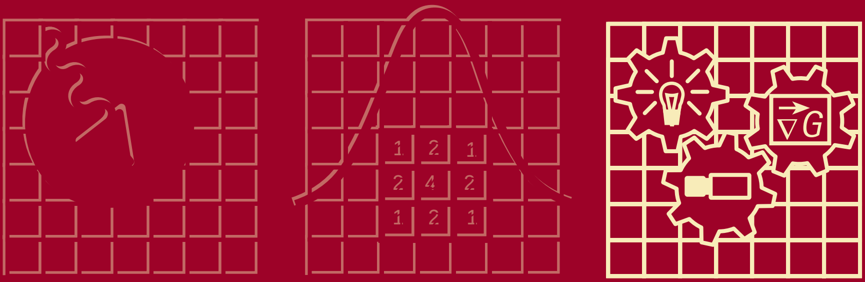


HANDBOOK OF COMPUTER VISION AND APPLICATIONS

Volume 3
Systems and Applications

Bernd Jähne
Horst Haußecker
Peter Geißler

ACADEMIC
PRESS

Handbook of Computer Vision and Applications

Volume 3
Systems and Applications

Handbook of Computer Vision and Applications

Volume 3
Systems and Applications

Editors
Bernd Jähne

Interdisciplinary Center for Scientific Computing
University of Heidelberg, Heidelberg, Germany
and
Scripps Institution of Oceanography
University of California, San Diego

Horst Haußecker
Peter Geißler

Interdisciplinary Center for Scientific Computing
University of Heidelberg, Heidelberg, Germany

ACADEMIC PRESS
San Diego London Boston
New York Sydney Tokyo Toronto

ACADEMIC PRESS
A Division of Harcourt Brace & Company
525 B Street, Suite 1900, San Diego, CA 92101-4495
http://www.apnet.com

ACADEMIC PRESS
24-28 Oval Road, London NW1 7DX, UK
http://www.hbuk.co.uk/ap/

Library of Congress Cataloging-In-Publication Data
Handbook of computer vision and applications / edited by Bernd Jähne, Horst Haussecker, Peter Geissler.
p. cm.
Includes bibliographical references and indexes.
Contents: v. 1. Sensors and imaging — v. 2. Signal processing and pattern recognition — v. 3. Systems and applications.
ISBN 0–12–379770–5 (set). — ISBN 0–12–379771-3 (v. 1)
ISBN 0–12–379772–1 (v. 2). — ISBN 0–12–379773-X (v. 3)
1. Computer vision — Handbooks, manuals. etc. I. Jähne, Bernd 1953– . II. Haussecker, Horst, 1968– . III. Geissler, Peter, 1966– .
TA1634.H36 1999
006.3′7 — dc21 98–42541
CIP

Printed in the United States of America
99 00 01 02 03 DS 9 8 7 6 5 4 3 2 1

Contents

Preface

What this handbook is about

This handbook offers a fresh approach to computer vision. The whole vision process from image formation to measuring, recognition, or reacting is regarded as an integral process. Computer vision is understood as the host of techniques to acquire, process, analyze, and understand complex higher-dimensional data from our environment for scientific and technical exploration.

In this sense the handbook takes into account the interdisciplinary nature of computer vision with its links to virtually all natural sciences and attempts to bridge two important gaps. The first is between modern physical sciences and the many novel techniques to acquire images. The second is between basic research and applications. When a reader with a background in one of the fields related to computer vision feels he has learned something from one of the many other facets of computer vision, the handbook will have fulfilled its purpose.

The handbook comprises three volumes. The first volume, *Sensors and Imaging*, covers image formation and acquisition. The second volume, *Signal Processing and Pattern Recognition*, focuses on processing of the spatial and spatiotemporal signal acquired by imaging sensors. The third volume, *Systems and Applications*, describes how computer vision is integrated into systems and applications.

Prerequisites

It is assumed that the reader is familiar with elementary mathematical concepts commonly used in computer vision and in many other areas of natural sciences and technical disciplines. This includes the basics of set theory, matrix algebra, differential and integral equations, complex numbers, Fourier transform, probability, random variables, and graphing. Wherever possible, mathematical topics are described intuitively. In this respect it is very helpful that complex mathematical relations can often be visualized intuitively by images. For a more formal treatment of the corresponding subject including proofs, suitable references are given.

How to use this handbook

The handbook has been designed to cover the different needs of its readership. First, it is suitable for *sequential reading*. In this way the reader gets an up-to-date account of the state of computer vision. It is presented in a way that makes it accessible for readers with different backgrounds. Second, the reader can look up specific topics of interest. The individual chapters are written in a self-consistent way with extensive cross-referencing to other chapters of the handbook and external references. The CD that accompanies each volume of the handbook contains the complete text of the handbook in the Adobe Acrobat portable document file format (PDF). This format can be read on all major platforms. Free Acrobat reader version 3.01 for all major computing platforms is included on the CDs. The texts are hyperlinked in multiple ways. Thus the reader can collect the information of interest with ease. Third, the reader can delve more deeply into a subject with the material on the CDs. They contain additional reference material, interactive software components, code examples, image material, and references to sources on the Internet. For more details see the readme file on the CDs.

Acknowledgments

Writing a handbook on computer vision with this breadth of topics is a major undertaking that can succeed only in a coordinated effort that involves many co-workers. Thus the editors would like to thank first all contributors who were willing to participate in this effort. Their cooperation with the constrained time schedule made it possible that the three-volume handbook could be published in such a short period following the call for contributions in December 1997. The editors are deeply grateful for the dedicated and professional work of the staff at AEON Verlag & Studio who did most of the editorial work. We also express our sincere thanks to Academic Press for the opportunity to write this handbook and for all professional advice.

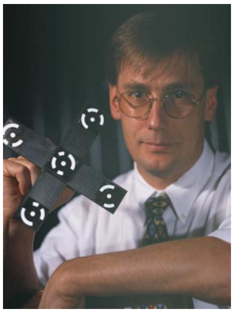
Last but not least, we encourage the reader to send us any hints on errors, omissions, typing errors, or any other shortcomings of the handbook. Actual information about the handbook can be found at the editors homepage `http://klimt.iwr.uni-heidelberg.de`.

Heidelberg, Germany and La Jolla, California, December 1998
Bernd Jähne, Horst Haußecker, Peter Geißler

Contributors

John Barron graduated from the University of Toronto, Ontario, Canada with an M.Sc. and PhD in Computer Science in 1980 and 1988, respectively. He is currently an associate professor in Computer Science at the University of Western Ontario. His research interests are in Computer Vision and include the measurement and interpretation of both optical and range flow and tracking (deformable) objects in long image sequences.

John Barron, Dept. of Computer Science
Middlesex College, The University of Western Ontario,
London, Ontario, N6A 5B7, Canada, barron@csd.uwo.ca



Horst A. Beyer graduated from the Swiss Federal Institute of Technology, Zurich, Switzerland with a Diploma and PhD in photogrammetry in 1982 and 1992, respectively and from the Ohio State University, Columbus, Ohio with a M.Sc. in Geodetic Science in 1985. He is founder and president of Imetric SA. His interests lie in vision based high accuracy three-dimensional measurements and camera calibration.

Horst A. Beyer, Imetric SA, Technopole
CH-2900 Porrentry, Switzerland
imetric@dial.eunet.ch, http://www.imetric.com

Maik Bollmann received his diploma in electrical engineering from the University of Paderborn in 1994. He has been a member of the IMA research group at the department of computer science, University of Hamburg since 1994. His research interests include attentive vision, models of visual attention, and color image processing. He is supported by the Deutsche Forschungsgemeinschaft.

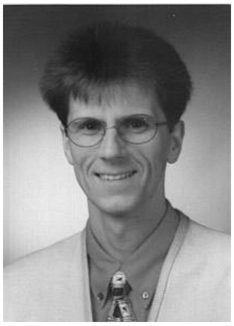
Dipl.-Ing. Maik Bollmann, University of Hamburg
Dept. of Computer Science, AG IMA
Vogt-Kölln-Str. 30, D-22527 Hamburg, Germany
bollmann@informatik.uni-hamburg.de
http://ima-www.informatik.uni-hamburg.de/˜bollmann

Harald Bornfleth studied physics at the Universities of Mainz and Heidelberg, both in Germany, where he received his diploma degree in May 1995. During this time, he spent a year at Glasgow University, Great Britain, with an Erasmus grant from the European Union. From September 1995 to July 1998, he worked on his PhD thesis at the Institute of Applied Physics and the Interdisciplinary Centre of Scientific Computing, University of Heidelberg. The thesis centered on the development of algorithms to analyze three-dimensional and four-dimensional (3D+time) patterns formed by replication-labeled DNA foci in cell nuclei.

Dr. Harald Bornfleth, Institut für Angewandte Physik, Universität Heidelberg
Albert-Überle-Str. 3-5, D-69120 Heidelberg, Germany
Harald.Bornfleth@IWR.Uni-Heidelberg.de

Katharina Braun received her diploma in biology and chemistry from Darmstadt University of Technology in 1980. From 1981 to 1987 she was research assistant at the Institute for Zoology at Darmstadt University, where she received her PhD in 1986. From 1988 to 1990 she was a postdoctoral fellow at the University of Washington. In 1991/92 she held a Helmholtz fellowship awarded from the German Ministry for Science and Technology (BMBF). She became leader of an independent research group in 1993. In 1994 she received her habilitation for zoology from Darmstadt University.

Katharina Braun, Leibniz Institute for Neurobiology
Brenneckestr. 6, 39118 Magdeburg, Germany
braun@ifn-magdeburg.de
http://www.ifn-magdeburg.de



Ulrich Büker studied computer science and mathematics at the University of Paderborn and received his diploma in 1990. He then joined the computer vision group in Paderborn and got his doctoral degree in electrical engineering in 1995. Currently he holds the position of an Oberingenieur in Paderborn. His main research interests are active vision systems, knowledge-based and neural recognition strategies for hybrid systems, and the use of parallel and distributed computing for the development of realtime vision systems.

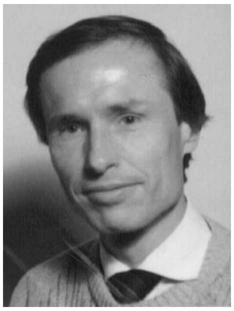
Dr.-Ing. Ulrich Büker, Heinz Nixdorf Institute
Department of Electrical Engineering, University of Paderborn
Pohlweg 47-49, D-33098 Paderborn, Germany
bueker@get.uni-paderborn.de
http://getwww.uni-paderborn.de/~bueker

Carlos Cárdenas holds a Dipl. Med. Inform. and is a PhD student in the basic research group. He has been a member of the medical IP group since 1996. His interests include graphical user interfaces, object-oriented development, and software ergonomics. He is a member of the IEEE and the BVMI.

Carlos E. Cárdenas S.
Div. Medical and Biological Informatics
Deutsches Krebsforschungszentrum
Im Neuenheimer Feld 280, D-69120 Heidelberg
http://mbi.dkfz-heidelberg.de/mbi/people/carlos.html

David Cheng recently completed both his Bachelor and Master degrees in Computer Science at the University of Western Ontario. He is currently a research programmer for the nuclear medicine department at the London Health Sciences Centre.

David Cheng, Dept. of Nuclear Medicine
The University of Western Ontario
London, Ontario, N6A 5B7, Canada
cheng@csd.uwo.ca



Christoph Cremer received degrees in Physics (University of Munich, 1970), Biology (University of Freiburg/Breisgau, 1976), and Human Genetics (University of Freiburg/Breisgau, 1983). Since October 1983 he has been Professor of Applied Optics and Information Processing at the University of Heidelberg, Faculty of Physics (Institute of Applied Physics). From 1970–1979 he worked at the Institute of Human Genetics, University of Freiburg; from 1980–1983 he worked at Lawrence Livermore National Laboratory, California as a visiting scientist. His present research interests are concentrated on the study of the 3-D structure of the human cell nucleus and its dynamics, using advanced methods of multispectral molecular fluorescence labeling, laser-supported light microscopy including spectral distance precision measurement procedures, and multidimensional image processing tools.

Prof. Dr. Christoph Cremer, Institute of Applied Physics
University of Heidelberg, Albert-Überle-Str. 3-5, D-69120 Heidelberg, Germany
cremer@popeye.aphys2.uni-heidelberg.de

Stefan Dauwe graduated in 1997 from the University of Heidelberg with a Master degree in physics. Since 1998 he has been working at the Institute for Solar Energy Research (ISFH), Hameln/Emmerthal, Germany. Now pursuing his PhD, he is working on the development of new, low-priced, crystalline Si solar cells.

Stefan Dauwe
Institut für Solarenergieforschung GmbH
Am Ohrberg 1, 31860 Emmerthal, Germany
S.Dauwe@isfh.de

Athanasios Demiris holds a Dipl. Med. Inform. and a PhD in medical informatics. He has been active in the area of object-oriented modeling since 1989 and a member of the medical IP group since 1992. His interests involve object-oriented information architectures, image analysis, fuzzy logic, and cognition-based software ergonomics. He was involved in the HELIOS II AIM project and the "Computer-aided Liver Resection Planning" project (funded by the Tumorzentrum Heidelberg/Mannheim). He is a member of the IEEE and the GI.

Athanasios M. Demiris
Div. Medical and Biological Informatics, Deutsches Krebsforschungszentrum
Im Neuenheimer Feld 280, D-69120 Heidelberg, Germany
a.m.demiris@dkfz-heidelberg.de
http://mbi.dkfz-heidelberg.de/mbi/people/thanos.html

E. D. Dickmanns studied aerospace engineering at the RWTH Aachen from 1956-1961 and control engineering at Princeton University, NJ from 1964-1965. From 1961–1975 he was a researcher with the German Aerospace Research Organisation DFVLR. He received his PhD in engineering from RWTH Aachen in 1969 and was a Postdoctoral Research Associate with NASA at the MSFC in Huntsville, AL, U.S. Since 1975 he has been full professor for control engineering at the University of the Federal Armed Forces of Germany, Munich (UniBwM), Department of Aero-Space Technology. Founder of the Institut für Systemdynamik und Flugmechanik (ISF) He was visiting professor at the California Institute of Technology in spring 1996 and at the Massachusetts Institute of Technology in fall 1998. His research subjects include dynamic machine vision; applications to road vehicle guidance, navigation of autonomously guided vehicles on the factory floor, landing approach of aircraft, landmark navigation for helicopters, and robotics in space.

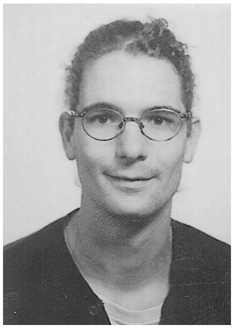
Prof. Dr. Ernst Dieter Dickmanns
Universität der Bundeswehr, München, D-85577 Neubibert, Germany
Ernst.Dickmanns@unibw-muenchen.de

Siegbert Drüe received his diploma and doctoral degree in electrical engineering from the University of Paderborn in 1983 and 1988. Since 1988 he has been Akademischer Oberrat at the Department of Electrical Engineering, University of Paderborn. His research interests include computer vision, active vision systems, and artificial neural networks and their applications.

Dr.-Ing. Siegbert Drüe, Heinz Nixdorf Institute
Department of Electrical Engineering
University of Paderborn
Pohlweg 47-49, D-33098 Paderborn, Germany
druee@get.uni-paderborn.de
http://getwww.uni-paderborn.de/~druee

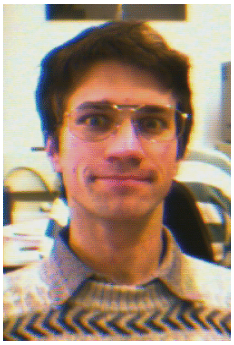
Peter Ulrich Edelmann studied physics at the University of Ulm, Germany from October 1990 to October 1992. Since then he has studied at the University of Heidelberg, specializing in commercial information technology and digital image processing. He received the diploma degree in physics from Heidelberg University in December of 1996. During his diploma thesis he worked on 3-D image analysis and reconstruction of the morphology of interphase chromosomes. Since January 1, 1997 he has been working on his PhD dissertation “Spectral Precision Distance Microscopy” in the Applied Optics and Information Processing group of Prof. C. Cremer.

Peter Ulrich Edelmann
Institut für Angewandte Physik, Universität Heidelberg
Albert-Überle-Str. 3-5, D-69120 Heidelberg, Germany
edelmann@popeye.aphys2.uni-heidelberg.de



Sven Eichkorn studied physics in Heidelberg, Germany and Santiago de Compostella, Spain. In 1997 he received his MSc degree. His thesis dealt with a LIF method to measure gas exchange. Currently he is doing research to acquire a PhD degree at the Max-Planck-Institute for Nuclear Physics, Heidelberg. His research interests include atmospheric trace gas physics.

Sven Eichkorn, Atmospheric Physics Division, Max-Planck-Institute for Nuclear Physics, Saupfercheckweg 1
D-69117 Heidelberg, Germany
Phone: +49 6221 516-0, Fax: +49 6221 516 324
Sven.Eichkorn@mpi-hd.mpg.de



Dirk Engelmann studied physics at the Technical University of Darmstadt, Germany. He is currently pursuing his PhD Thesis at the Interdisciplinary Center for Scientific Computing at Heidelberg University. His research topic is flow dynamics close air/water interface and his research interests include measurements of the flow field near the free water surface with the aid of 4-D Particle Tracking Velocimetry and numerical simulation of flow applying numerical finite element solvers on Navier Stokes partial differential equations.

Dirk Engelmann
Forschungsgruppe Bildverarbeitung, IWR
Universität Heidelberg, Im Neuenheimer Feld 368
D-69120 Heidelberg, Germany
Dirk.Engelmann@iwr.uni-heidelberg.de
http://klimt.iwr.uni-heidelberg.de/˜dengel

Rainer H.A. Fink is a professor at the II. Institute of Physiology at the University of Heidelberg. His research interests comprise calcium regulation, activation of contractile force, membrane electrophysiology, and laser applications in the biophysics of muscular contraction. He held research and teaching positions at the University of Washington, Seattle, WA, U.S., La Trobe University, Melbourne, and the University of Adelaide, Australia, before taking up his professorship in Heidelberg in 1990. He received his PhD in 1979 at the University of Bochum, Germany.

Prof. Dr. Rainer H.A. Fink, II. Physiologisches Institut
Universität Heidelberg, Im Neuenheimer Feld 326
D-69120 Heidelberg, Germany
fink@novsrv1.pio1.uni-heidelberg.de

Stefan Fries studied physics at the Universities of Karlsruhe and Augsburg, Germany. He worked at the Max-Planck-Institute for Plasma Physics in Garching, Germany on simulations in the field of electromagnetism. In 1996, he earned a diploma degree in physics from the University of Augsburg. Since 1997 he has been a member of the research group on algorithm assessment at the Fraunhofer Institute for Information and Data Processing in Karlsruhe. His interests are the investigation of algorithm performance characteristics and the development of software tools to measure these quantities.

Dipl.-Phys. Stefan Fries
FhG-Institut für Informations- und Datenverarbeitung
Fraunhoferstr. 1, D-76131 Karlsruhe, Germany, fri@iitb.fhg.de

Robert Frischholz studied computer science in Erlangen. Since 1991 he was working for the Fraunhofer Institute IIS in the field of software development for high-speed cameras and motion analysis systems. From 1996 to 1998, he was the leader of the development department of Mikromak GmbH. In 1998 he received his doctoral degree from Erlangen University. Since August 1998, he is the development manager of DCS AG, Berlin-Erlangen, and is currently involved in research and development of biometric person identification.

Dr. Robert Frischholz, DCS AG, Wetterkreuz 19a
D-91058 Erlangen, Germany
frz@dcs.de, http://www.bioid.com

Christoph Sebastian Garbe studied physics at the University of Hamburg, Germany and the University of Heidelberg, Germany and is currently pursuing his Diploma Thesis at the Interdisciplinary Center for Scientific Computing and the Institute for Environmental Physics in Heidelberg. His research interests include measurements of the flow fields near the free water surface with the aid of 4-D Particle Tracking Velocimetry.

Christoph Sebastian Garbe
Forschungsgruppe Bildverarbeitung
Interdisciplinary Center for Scientific Computing
Im Neuenheimer Feld 368, D-69120 Heidelberg, Germany
Christoph.Garbe@iwr.uni-heidelberg.de
http://klimt.iwr.uni-heidelberg.de

Peter Geißler studied physics in Heidelberg. He received his diploma and doctoral degree from Heidelberg University in 1994 and 1998, respectively. His research interests include computer vision, especially depth-from-focus, adaptive filtering, and flow visualization as well as the application of image processing in physical sciences and oceanography.

Dr. Peter Geißler
Forschungsgruppe Bildverarbeitung, IWR
Universität Heidelberg, Im Neuenheimer Feld 368
D-69120 Heidelberg, Germany
Peter.Geissler@iwr.uni-heidelberg.de
http://klimt.iwr.uni-heidelberg.de

Matthias Graf studied physics at the University of Karlsruhe from 1991 to 1998. Between 1995 and 1997 he developed the topographical measurement system for layer thickness at the Fraunhofer Institut für Chemische Technologie (ICT) in Pfinztal, Germany. Since 1998 he has been working on his doctorate in the microwave processing group at the Institut für Kunststoffprüfung und Kunststoffkunde (IKP), University of Stuttgart, Germany.

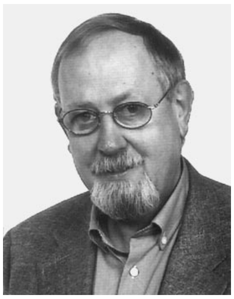
Matthias Graf
Institut für Kunststoffprüfung und Kunststoffkunde (IKP), Pfaffenwaldring 32, D-70569 Stuttgart, Germany
graf@ikp.uni-stuttgart.de, Matthias.Graf@t-online.de
http://www.ikp.uni-stuttgart.de

Hermann Gröning graduated in 1996 from the University of Heidelberg with a master degree in physics and is now pursuing his PhD at the Interdisciplinary Center for Scientific Computing. He is concerned mainly with radiometric and geometric camera calibration.

Hermann Gröning
Forschungsgruppe Bildverarbeitung, IWR
Universität Heidelberg
Im Neuenheimer Feld 368
D-69120 Heidelberg, Germany
Hermann.Groening@iwr.uni-heidelberg.de

Gerd Häusler is adjunct professor, University of Erlangen, Chair for Optics, and director of the Optical Metrology Group. He received his diploma in 1970 and a doctoral degree in 1974 from the Optical Institute, Technical University Berlin. In 1974 he moved to the Chair for Applied Optics (later Chair for Optics), University of Erlangen. There he received his habilitation in 1982. As a doctoral fellow he worked with IBM (Sindelfingen), ENST Telecom (Paris), and RCA (Zürich). At the University of Munich and the RIKEN Institute in Tokyo he worked on optical and electronical image processing and nonlinear optical feedback systems. His current research interests include the investigation of the physical limits of range sensing and the construction of sensors that work at these limits and cover the nanometer to meter range, with applications in industry and medicine.

Prof. Dr. Gerd Häusler, Chair for Optics, Universität Erlangen-Nürnberg
Staudtstraße 7/B2, D-91056 Erlangen, Germany
haeusler@physik.uni-erlangen.de



Georg Hartmann received his M.S. degree in physics from the University of Erlangen (1962) and after postgraduate studies of nuclear physics he received his PhD degree (1968). He was development engineer in the field of nuclear instrumentation, radiometric measurement, and industrial automation between 1968 and 1976, and head of development until 1979. Since then he joined the faculty of Electrical Engineering at the University of Paderborn and has been a member of the governing board of the Heinz Nixdorf Institut, a center of interdisciplinary research in the field of computer science at Paderborn University. Between 1983 and 1987 he served as Vice President at this university. His field of research is computer vision, and since 1994 he has been President of the German Association for Pattern Recognition (DAGM).

Prof. Dr. Georg Hartmann, Heinz Nixdorf Institut
Universität Paderborn, Pohlweg 47-49, D-33098 Paderborn, Germany
hartmann@get.uni-paderborn.de
http://getwww.uni-paderborn.de/~hartmann

Michael Hausmann received his Diploma degree in Physics in 1984 and his PhD in 1988 (University of Heidelberg). After his habilitation in 1996 he became deputy-leader of the research division “Applied Optics and Information Processing” at the Institute of Applied Physics, University of Heidelberg. His field of research is centered around multi-dimensional flow cytometry, high-resolution light microscopy, digital image analysis, and multi-parameter immunobiological labeling techniques to study high-order chromatin structures.

Dr. Michael Hausmann
Institute of Applied Physics, University of Heidelberg
Albert-Überle-Str. 3-5, 69120 Heidelberg, Germany
hausmann@popeye.aphys2.uni-heidelberg.de

Thorsten Hermes received the B.Sc. degree in computer science, in 1989, and the M.Sc. degree (Diplom) in computer science from the University of Hamburg, Germany, in 1994. Currently he is working towards his PhD (texture analysis) as a research scientist at the Center for Computing Technology at the University of Bremen. His research interests include computer vision, video analysis, neural networks, biological background of (human) vision, and content-based image retrieval.

Dipl.-Inform. Thorsten Hermes,
Center for Computing Technology
Image Processing Department
University of Bremen, P.O. Box 33 0440, D-28334 Bremen, Germany
hermes@tzi.org, http://www.tzi.org/~hermes

Horst Haußecker studied physics in Heidelberg. He received his diploma in physics and his doctoral degree from Heidelberg University in 1994 and 1996, respectively. He was visiting scientist at the Scripps Institution of Oceanography in 1994. Currently he is conducting research in the image processing research group at the Interdisciplinary Center for Scientific Computing (IWR), where he also lectures on optical flow computation. His research interests include computer vision, especially image sequence analysis, infrared thermography, and fuzzy-image processing, as well as the application of image processing in physical sciences and oceanography.

Dr. Horst Haußecker, Forschungsgruppe Bildverarbeitung, IWR
Universität Heidelberg, Im Neuenheimer Feld 368, D-69120 Heidelberg
Horst.Haussecker@iwr.uni-heidelberg.de
http://klimt.iwr.uni-heidelberg.de

Frank Hering studied physics at the technical university of Clausthal and at the university of Heidelberg. He received his diploma and doctoral degree from Heidelberg University in 1993 and 1997. From 1993 until 1998 he headed the flow visualization team in Prof. Jähne's research group at the Interdisciplinary Research Center for Scientific Computing, Heidelberg. His research interests include computer vision, image sequence analysis, and flow visualization. In 1998 he left the university and is now a member of the coordination team for logistics development at the SAP AG, Walldorf, Germany.

Dr. Frank Hering, Forschungsgruppe Bildverarbeitung, IWR
Universität Heidelberg, Im Neuenheimer Feld 368, D-69120 Heidelberg
Frank.Hering@iwr.uni-heidelberg.de or Frank.Hering@sap-ag.de

Helmut Herrmann studied electrical engineering at the Technische Hochschule Darmstadt, Germany, and received his diploma in 1981. After some years in the industry, he began his study of computer sciences at the FernUniversität Hagen, Germany, and received his diploma in 1996. Since 1992 he has been working on the development team of the image processing software heurisko.

Helmut Herrmann, Department for Image Processing
AEON Verlag & Studio Walter H. Dorn
Fraunhoferstraße 51 B, D-63454 Hanau, Germany
Helmut.Herrmann@aeon.de, http://www.aeon.de

Otthein Herzog holds a chair in the Department of Mathematics and Computer Science at the University of Bremen, where he is the head of the Artificial Intelligence Research Group and director of the Center for Computing Technologies, a software-oriented technology transfer center within the University that he founded in 1995. He graduated in Applied Mathematics at the University of Bonn, Germany in 1972 and received his PhD from the Computer Science Department at the University of Dortmund in 1976. From 1977 through 1993 he worked for IBM Germany in operating and communications systems development, software engineering, and in research. From 1988 through 1992 he directed the Institute for Knowledge-Based Systems in the IBM Germany Science Center. From 1992 to 1993 he was a Senior Manager in the IBM Germany Software Development Laboratory. His current research interests include digital libraries, knowledge-based image and video processing, agent-base software technologies, and the transfer of these technologies into real-world applications in the areas of multimedia, intelligent environmental systems, and logistics.

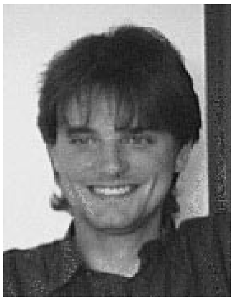
Prof. Dr. Otthein Herzog
Center for Computing Technology — Image Processing Department
University of Bremen, P.O. Box 33 0440, D-28334 Bremen, Germany
herzog@tzi.org, http://www.tzi.org/~herzog

Wolfgang Hilberg received his diploma in electrical engineering, especially communications and high frequency. From 1958 to 1963 he was research assistant at the Telefunken Research Institute in Ulm, Germany. He received his PhD in 1963 at Darmstadt University of Technology. Since 1972 he has been a professor at Darmstadt University.

Prof. Dr. Wolfgang Hilberg, Fachgebiet Digitaltechnik
Fachbereich 18, Elektrotechnik und Informationstechnik
Technische Hochschule Darmstadt
Merckstr. 25, D-64283 Darmstadt, Germany
hil@dtro.tu-darmstadt.de

Rainer Hoischen received his diploma in electrical engineering from University of Paderborn in 1993. He has been a member of the IMA research group at the department of computer science, University of Hamburg since 1994. His research interests include motion detection, motion estimation, and object tracking with active vision systems.

Dipl.-Ing. Rainer Hoischen
Universität Hamburg
Fachbereich Informatik, AG IMA
Vogt-Kölln-Str. 30, D-22527 Hamburg, Germany
hoischen@informatik.uni-hamburg.de
http://ima-www.informatik.uni-hamburg.de/~hoischen



Joachim Hornegger graduated in 1992 and received his PhD degree in computer science in 1996 from the Universität Erlangen-Nürnberg, Germany, for his work on statistical object recognition. Joachim Hornegger was research and teaching associate at Universität Erlangen-Nürnberg, a visiting scientist at the Technion, Israel, and at the Massachusetts Institute of Technology, U.S. He is currently a research scholar and teaching associate at Stanford University, U.S. Joachim Hornegger is the author of 30 technical papers in computer vision and speech processing and three books. His research interests include 3-D computer vision, 3-D object recognition, and statistical methods applied to image analysis problems.

Dr. Joachim Hornegger, Stanford University, Robotics Laboratory
Gates Building 1A, Stanford, CA 94305-9010, U.S.
jh@robotics.stanford.edu, http://www.robotics.stanford.edu/˜jh

Bernd Jähne studied physics in Saarbrücken and Heidelberg. He received his diploma, doctoral degree, and habilitation degree from Heidelberg University in 1977, 1980, and 1985, respectively, and a habilitation degree in applied computer science from the University of Hamburg-Harburg in 1992. Since 1988 he has been a Marine Research Physicist at Scripps Institution of Oceanography, University of California, and, since 1994, he has been professor of physics at the Interdisciplinary Center of Scientific Computing. He leads the research group on image processing. His research interests include computer vision, especially filter design and image sequence analysis, the application of image processing techniques in science and industry, and small-scale air-sea interaction processes.

Prof. Dr. Bernd Jähne, Forschungsgruppe Bildverarbeitung, IWR
Universität Heidelberg, Im Neuenheimer Feld 368, D-69120 Heidelberg
Bernd.Jaehne@iwr.uni-heidelberg.de
http://klimt.iwr.uni-heidelberg.de

Paul Joe has a B.Ap.Sc. and Eng., a M.Sc. and a PhD in physics from the University of Toronto. He subsequently joined the Cloud Physics Research Division of Atmospheric Environment Service of Canada. He began work in Doppler weather radar in 1985 at the King City Weather Radar station. He is an expert in meteorological instrumentation, cloud physics, weather radar, and severe weather. He is currently a research scientist and acting chief of the Cloud Physics Research Division, chair of the Radar Committee of the American Meteorological Society, software project manager for the National Radar Project (Canada) and interim chair of the Nowcasting Demonstration Project of the World Weather Research Program.

Paul Joe, King City Radar Station, Atmospheric Environmental Services
4905 Dufferin St., Toronto, Ontario M3H 5T4, Canada
joep@aestor.dots.doe.ca

Stefan Karbacher received his PhD in physics from the Chair for Optics of the University of Erlangen-Nürnberg. He currently works as development engineer in the field of 3-D image processing, surface reconstruction, and scattered data modeling.

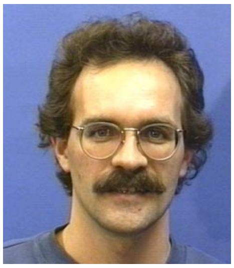
Stefan Karbacher, Chair for Optics
University of Erlangen-Nürnberg
Staudtstraße 7/B2
D-91056 Erlangen, Germany
sbk@undine.physik.uni-erlangen.de
www.physik.uni-erlangen.de/optik/haeusler/people/sbk/sbk_home_e.html

Christoph Klauck received his diploma in computer science and mathematics from the University of Kaiserslautern, Germany, in 1990. From 1990 to 1994 he worked as research scientist at the German Research Center for Artificial Intelligence Inc. (DFKI GmbH) at Kaiserslautern. In 1994 he finished his dissertation in computer science. Since then he has been involved in the IRIS project at the University of Bremen (Artificial Intelligence Group). His primary research interests include graph grammars and rewriting systems in general, knowledge representation, and ontologies.

Prof. Dr. Christoph Klauck, Dep. of Electrical Eng. and Computer Science
University of Hamburg (FH), Berliner Tor 3, D-20099 Hamburg, Germany
cklauck@t-online.de, http://fbi010.informatik.fh-hamburg.de/~klauck

Peter Klausmann studied technomathematics at the Universities of Kaiserslautern, Germany, and Grenoble, France. Since 1995 he has worked at the Fraunhofer-Institute for Information and Data Processing in Karlsruhe. In 1995-1997 he worked in a research project concerned with image sequence analysis for automatic vehicle guidance. Since 1997 he has been a member of the research group on algorithm assessment. His research interests include the development, evaluation, and assessment of algorithms for automatic target recognition.

Dipl.-Math. techn. Peter Klausmann
Fraunhofer-Institut für Informations- und Datenverarbeitung
Fraunhoferstr. 1, D-76131 Karlsruhe, Germany, klm@iitb.fhg.de



Reinhard Koch received a M.Sc. in physical science from FHSU University, Hays, KS, U.S. in 1982 and a diploma in electrical engineering from the University of Hannover, Germany in 1985. He took a position as research assistant at the Institute of Theoretical Information Processing of the University of Hannover in 1988 and obtained a Ph.D. in electrical engineering in 1996. Since 1996 he has been working as computer vision researcher at the Department of Electrical Engineering of the Katholieke Universiteit Leuven, Belgium. His research interests include computer vision and extraction of 3-D scene structure from image sequences. For his research on 3-D modeling from stereoscopic image sequences he received the Olympus Award 1997, and together with Colleagues in Leuven and Oxford he received the David Marr Award in 1998 for work on 3-D reconstruction from uncalibrated image sequences.

Dr. Reinhard Koch, Katholieke Universiteit Leuven, Dept. Elektrotechniek
ESAT-PSI, Kardinaal Mercierlaan 94, B-3001 Leuven, Belgium
Reinhard.Koch@esat.kuleuven.ac.be
http://www.esat.kuleuven.ac.be/~koch

Ullrich Köthe received the diploma degree in physics from the University of Rostock, Germany. Currently he is a researcher at the Fraunhofer Institute for Computer Graphics in Rostock. He has worked in the field of image processing and computer vision since 1992. His main research interests are image segmentation and 3-D object reconstruction where he authored or co-authored a number of papers. Besides this, he has devoted significant attention to the problem of reusable implementation of computer vision software and developed several new implementation techniques based on the generic programming paradigm.

Ullrich Köthe, Fraunhofer-Institut für Graphische Datenverarbeitung
Joachim-Jungius-Str. 11, D-18059 Rostock, Germany
phone +49 381 / 4024 114, fax +49 381 / 4024 199
koethe@egd.igd.fhg.de, http://www.egd.igd.fhg.de/~ulli

Bernd Kümmerlen graduated in 1998 from the University of Heidelberg with a master degree (Diplom) in physics on the topic “infrared thermography to study physiological parameters of plant leaves”. Currently he works as a research assistant at MeVis (Center for Diagnostic Systems and Visualization) in Bremen, Germany. Now pursuing his PhD, he is working on the development of a software tool for the analysis of dynamic medical CT and MR data.

Bernd Kümmerlen, MeVis
Center for Diagnostic Systems and Visualization
Universitätsallee 29, 28359 Bremen, Germany, bkuemmer@mevis.de

Reinhard Koy-Oberthür received the Dipl.-Ing. (M.Sc.) degree in electrical engineering from Technical University of Hannover in 1982 and the Dr.-Ing. (PhD) degree in electrical information technology from Technical University of Clausthal-Zellerfeld in 1987. He continued working on a research project on computer vision and mobility aids for blind people supported by the DFG. Since 1989, he has been with VITRONIC, Wiesbaden where he is manager of OCR identification technologies.

Reinhard Koy-Oberthür
VITRONIC Dr.-Ing. Stein Bildverarbeitungssysteme GmbH
Hasengartenstr. 14a, D-65189 Wiesbaden, Germany
Phone: +49-611-7152-29, rko@vitronic.de

Stefan Lanser received his diploma in computer science in 1992 and his doctoral degree in 1997 from the Technische Universität München. He is currently a research assistant at the Forschungsgruppe Bildverstehen FG BV (computer vision research group) at the same institution. In 1996, he was one of the founders of MVTec Software GmbH, the vendor of the image analysis tool HALCON. His research interests include image processing tools and their industrial applications, and CAD-based computer vision, especially model-based object recognition and pose estimation.

Dr. Stefan Lanser
Forschungsgruppe Bildverstehen (FG BV), Informatik IX
Technische Universität München, D-80290 München, Germany
Stefan.Lanser@in.tum.de
http://wwwradig.in.tum.de/people/lanser/

Ralf Lay is manager and cofounder of Silicon Software GmbH, Mannheim. During his PhD studies at the Lehrstuhl für Informatik V, University of Mannheim, Germany, he developed a high-level language for programming FPGA processor systems. He received his PhD and diploma in physics from the University of Heidelberg, Germany.

Dr. Ralf Lay, Silicon Software GmbH
M2, 16, D-68161 Mannheim, Germany
lay@silicon-software.com
http://www.silicon-software.com

Carsten Leue studied physics in Dortmund and Heidelberg and received a diploma in 1996. Within the GOME project he is working on his PhD thesis founded by the *Studienstiftung des Deutschen Volkes* in a collaboration between the Interdisciplinary Center for Scientific Computing and the Institute for Environmental Physics in Heidelberg.

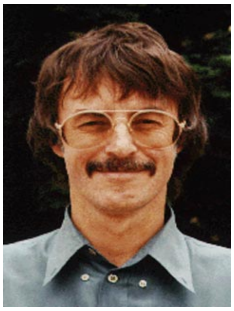
Carsten Leue, Institut für Umweltphysik
Universität Heidelberg, Im Neuenheimer Feld 366
D-69120 Heidelberg, Germany
carsten.leue@iwr.uni-heidelberg.de
http://klimt.iwr.uni-heidelberg.de/˜cleue

Ulrike Lode studied physics in Heidelberg, Germany, and San Diego, California. She is currently doing research at the Institute for Environmental Physics in Heidelberg to develop a new LIF-technique to measure air/water gastransfer. Her research interests include spectrometry, gas exchange, and LIF.

Ulrike Lode
Institut für Umweltphysik
Universität Heidelberg
Im Neuenheimer Feld 366
69120 Heidelberg, Germany
ulode@giotto.iwr.uni-heidelberg.de

Markus Loose is a PhD student at Heidelberg University, where he is involved in research on vision chip camera systems. He studied physics in Konstanz and Heidelberg and received his diploma in physics from Heidelberg University in 1996.

Markus Loose, IHEP
Universität Heidelberg, Schröderstrasse 90
D-69120 Heidelberg, Germany
loose@asic.uni-heidelberg.de
http:www.ihep.uni-heidelberg.de/˜loose

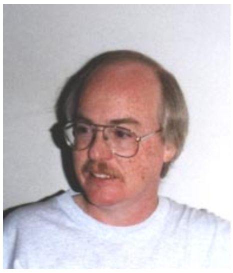


Reinhard Männer is head of the Department for Computer Sciences V, University of Mannheim, and member of the Interdisciplinary Center for Scientific Computing, University of Heidelberg. His research interests are reconfigurable (FPGA-based) processors, biomedical image processing and simulation, and special purpose processors for computationally demanding applications like real-time volume visualization and high-speed pattern recognition. He is cofounder of the companies Heidelberg Instruments, Medical Communications, Volume Graphics, and Silicon Software. Männer received his diploma in physics in 1974 from the University of Munich, his doctoral degree in 1979, and his habilitation degree in 1986 from the University of Heidelberg.

Prof. Dr. Reinhard Männer, Lehrstuhl für Informatik V
Universität Mannheim, B6, 26, D-68131 Mannheim, Germany
maenner@ti.uni-mannheim.de, http://www.informatik.uni-mannheim.de

Hans Peter Meinzer studied physics and economics at Karlsruhe University and obtained an MS in 1973. He received a doctorate (1983) and a habilitation (1987) in medical computer science from Heidelberg University. Today he is an associate professor at the Heidelberg University on medical computer science and director of a research team specializing in medical image processing and tissue kinetics simulation (since 1983). He won scientific awards from the German Society of Heart Surgery (1992), the German Society of Pattern Recognition (1993) and the European Commission (1997), the "European Information Technology Prize." Meinzer founded and directs the Steinbeis-Transferzentrum Medizinische Informatik (TZMI) in 1994, a software company specializing in telemedicine and teleradiology.

Prof. Dr. Hans Peter Meinzer, Div. Medical and Biological Informatics
Deutsches Krebsforschungszentrum (DKFZ)
Im Neuenheimer Feld 280, D-69120 Heidelberg
http://mbi.dkfz-heidelberg.de/mbi/people/pitt.html



Robert E. Mercer is associate professor in the Dept. of Computer Science at the University of Western Ontario. He is director of the Cognitive Engineering Laboratory and co-director of the Centre for Cognitive Science (Artificial Intelligence Group). He received his PhD in Computer Science from The University of British Columbia in 1987. His research interests include developing various knowledge representation frameworks and schemes and their associated reasoning mechanisms, and applying these processes to natural language, computer vision, animation, human-computer interaction, and information retrieval.

Prof. Robert E. Mercer, Dept. of Computer Science, Middlesex College
The University of Western Ontario, London, Ontario, N6A 5B7, Canada
mercer@csd.uwo.ca

Bärbel Mertsching is the head of the IMA research group. She received her diploma and PhD degrees in electrical engineering from the University of Paderborn. Since 1994, she has been a full professor at the University of Hamburg. After working on knowledge-based image understanding, she is now doing research that is contributing to the synthesis of symbolic computation and artificial neural nets for computer vision. Additionally, her current interests include parallel algorithms and hardware for computer vision and speech understanding. Her main goal is to put "intelligent" systems to work by integrating hard- and software components.

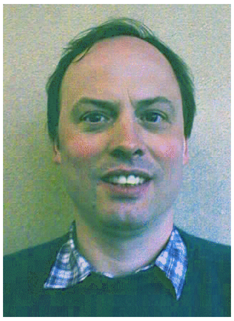
Prof. Dr.-Ing. Bärbel Mertsching, Universität Hamburg
Dept. of Computer Science, Vogt-Kölln-Str. 30, D-22527 Hamburg, Germany
mertsching@informatik.uni-hamburg.de
http://ima-www.informatik.uni-hamburg.de/˜mertschi

Bernhard Minge received the Dipl.-Ing. degree in mechanical engineering from the Technical University of Clausthal-Zellerfeld, Germany, in 1989. Since 1989 he has been with VITRONIC Dr.-Ing. Stein Bildverarbeitungssysteme GmbH in Wiesbaden, Germany. He started working with system engineering in the fields of robot vision and quality inspection. He is the 3-D systems manager.

Bernhard Minge, VITRONIC Dr.-Ing. Stein
Bildverarbeitungssysteme GmbH
Hasengartenstrasse 14a, D-65189 Wiesbaden, Germany
bm@vitromic.de

Thorsten Maucher is a diploma student in physics at Heidelberg University, working on tactile displays.

Thorsten Maucher, IHEP
Universität Heidelberg, Schröderstrasse 90
D-69120 Heidelberg, Germany
maucher@asic.uni-heidelberg.de



Karlheinz Meier has been a professor of physics at Heidelberg University since October 1992. He received his PhD in physics from Hamburg University 1984. From 1984–1986 he was a research fellow at the European Center for Nuclear Research (CERN), Geneva, from 1986–1990 a scientific staff member at CERN, and from 1990–1992 scientific staff member at the Deutsches Elektronen Synchrotron (DESY), Hamburg.

Prof. Dr. Karlheinz Meier, IHEP, Universität Heidelberg
Schröderstrasse 90, D-69120 Heidelberg, Germany
meierk@ihep.uni-heidelberg.de

Thomas Münsterer received his diploma (M.Sc.) in physics and his PhD in physics from the University of Heidelberg, Germany, in 1993 and 1996, respectively. His thesis work involved using laser-induced fluorescence techniques and image processing in the investigation of air-water mass transfer. Since 1996, he has been with Vitronic Dr.-Ing. Stein Bildverarbeitungssysteme GmbH in Wiesbaden, Germany. He has been engaged in system design in the OCR department and in the development of illumination techniques and low-level image processing routines.

Thomas Münsterer, VITRONIC Dr.-Ing. Stein Bildverarbeitungssysteme GmbH
Hasengartenstr. 14a, D-65189 Wiesbaden, Germany
Phone: +49-611-7152-38, email: tm@vitronic.de

Olaf Munkelt received his diploma in mathematics in 1989 at the University of Karlsruhe and his doctoral degree in 1994 from the Technische Universität München. He is currently co-head of the research group, Cognitive Systems, at the Bavarian Research Center for Knowledge-Based Systems (FORWISS). In 1996 he was one of the founders of MVTec Software GmbH, the vendor of the image analysis tool HALCON. His research interests include computer vision and object recognition, image understanding, and CAD-based vision. He also works on video applications for autonomous mobile systems.

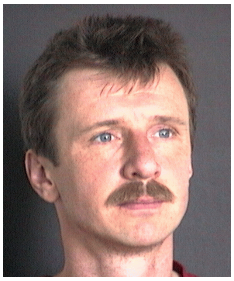
Dr. Olaf Munkelt, FORWISS, Bayerisches Forschungszentrum für Wissensbasierte Systeme, Forschungsgruppe Kognitive Systeme
Orleansstr. 34, 81667 München, Germany
munkelt@forwiss.de, http://www.forwiss.de/~munkelt

Heinrich Niemann obtained the degree of Dipl.-Ing. in electrical engineering and Dr.-Ing. at Technical University Hannover in 1966 and 1969, respectively. From 1967 to 1972 he was with Fraunhofer Institut für Informationsverarbeitung in Technik und Biologie, Karlsruhe. Since 1975 he has been professor of computer science at the University of Erlangen-Nürnberg and since 1988 he has also served as head of the research group, Knowledge Processing, at the Bavarian Research Institute for Knowledge-Based Systems (FORWISS). His fields of research are speech and image understanding and the application of artificial intelligence techniques in these fields. He is the author or co-author of 6 books and approximately 250 journal and conference contributions.

Prof. Dr.-Ing. H. Niemann, Lehrstuhl für Mustererkennung (Informatik 5)
Universität Erlangen-Nürnberg, Martensstraße 3, 91058 Erlangen, Germany
niemann@informatik.uni-erlangen.de
http://www5.informatik.uni-erlangen.de

Klaus-Henning Noffz is manager and cofounder of Silicon Software GmbH, Mannheim. During his PhD study at the Lehrstuhl für Informatik V, University of Mannheim, Germany, he developed large FPGA processor systems for application in high-energy physics. Dr. Noffz received his PhD and diploma in physics at the University of Heidelberg, Germany.

Dr. Klaus-Henning Noffz, Silicon Software GmbH
M2, 16, D-68161 Mannheim, Germany
noffz@silicon-software.com
http://www.silicon-software.com



Dietrich Paulus received a bachelor degree in computer science at the University of Western Ontario, London, Canada (1983). He graduated (1987) and received his PhD degree (1991) from the University of Erlangen-Nürnberg, Germany. He is currently a senior researcher (Akademischer Rat) in the field of image pattern recognition and teaches courses in computer vision and applied programming for image processing. Together with J. Hornegger, he has recently written a book on pattern recognition and image processing in C++.

Dr. Dietrich Paulus, Lehrstuhl für Mustererkennung
Universität Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen, Germany
paulus@informatik.uni-erlangen.de
http://www5.informatik.uni-erlangen.de

Peter Plankensteiner received a diploma degree in mathematics from the University of Munich, Germany, in 1993. He is currently finishing his PhD degree in the department of imaging at the Fraunhofer Institute for Integrated Circuits in Erlangen. Since 1994, he has been developing image and signal processing systems and since 1997 he has led the intelligent systems group. His research fields comprise industrial object recognition, automatic inspection of wood, and the understanding of biometrics including face, gesture, and speech.

Dipl.-Math. Peter Plankensteiner, Fraunhofer Institut für Integrierte Schaltungen, Am Weichselgarten 3, D-91058 Erlangen, Germany
ppl@iis.fhg.de, http://www.iis.fhg.de

Ulrich Platt introduced—together with D. Perner—the DOAS technique for the measurement of atmospheric trace gases. He has long-standing experience in the field of scattered-light DOAS measurements of stratospheric gases, in particular of free radicals and halogen species. He is a member of the science advisory committee of the GOME and SCIAMACHY satellite instrument. At present he is director of the Institute for Environmental Physics, University of Heidelberg.

Prof. Dr. Ulrich Platt, Institut für Umweltphysik
University of Heidelberg, Im Neuenheimer Feld 366, Germany
69120 Heidelberg, Germany, pl@uphys1.uphys.uni-heidelberg.de

Bernd Radig received his diploma in physics from the University of Bonn in 1973 and his doctoral degree in computer science from the University of Hamburg in 1978. In 1982 he received a habilitation degree in computer science from the latter institution. From 1982 to 1986 he was professor on the faculty of computer science, University of Hamburg, leading the research group on cognitive systems. Since then he has been professor and one of the directors of the institute for computer science at the Technische Universität München. Moreover, he has been director of the Bavarian Research Center for

Knowledge-Based Systems (FORWISS) since 1988 and since 1993 chairman of AbayFOR, a co-operative venture involving 24 Bavarian research institutes. He is head of the “Forschungsgruppe Bildverstehen (FG BV)” at the Technische Universität München and the research group, Cognitive Systems, at FORWISS. His research interests cover various fields in artificial intelligence, computer vision and image understanding, signal processing, and pattern recognition.

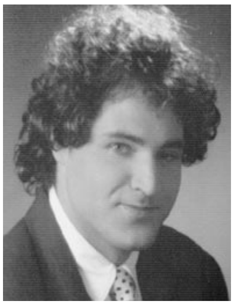
Prof. Dr. Bernd Radig, Informatik IX, Technische Universität München
80290 München, Germany, radig@in.tum.de, http://wwwradig.in.tum.de/

Christof Ridder finished his apprenticeship as an electrician in 1986. From 1988 to 1993 he studied computer science at the Fachhochschule München and received his diploma in machine vision/texture analysis. He is currently a PhD student with the research group, Cognitive Systems, at the Bavarian Research Center for Knowledge-Based Systems (FORWISS). His main research interests are model-based object recognition for video surveillance tasks and active vision.

Christof Ridder, FORWISS, Bayerisches Forschungszentrum für Wissensbasierte Systeme
Forschungsgruppe Kognitive Systeme
Orleansstr. 34, 81667 München, Germany
ridder@forwiss.de, http://www.forwiss.de/˜ridder

Torsten Scheuermann studied mechanical engineering at the University of Karlsruhe where he specialized in optical measurement and aerodynamics. At the Fraunhofer Institut für Chemische Technologie (ICT) in Pfinztal, Germany, he led an R&D group from 1993 through 1996 developing new approaches for digital visualization and optical measurement. In 1997 he obtained his doctorate for a thesis about a new optical measurement system. He then moved to the Institute for Polymer Science and Polymer Testing (IKP) at the University of Stuttgart and to PE Product Engineering GmbH (Dettingen, Germany). Since January 1998 he has been working for Fraunhofer USA, Ann Arbor, MI, and represents the Fraunhofer ICT and its partners IKP and PE in the U.S.

Dr. Torsten Scheuermann, Fraunhofer USA, Headquarters
24 Frank Lloyd Wright Drive, Ann Arbor, MI 48106-0335, U.S.
tscheuermann@fraunhofer.org, http://www.fraunhofer.org



Uwe Udo Schimpf is graduate researcher in the research unit “Image Sequence Analysis to Investigate Dynamic Processes,” subproject B “Wind waves, turbulence and exchange processes at the ocean surface.” He was development technician at the Scripps Institution of Oceanography, University of California, San Diego, where he participated in two research cruises. His master thesis was on microscale temperature fluctuations at the water surface. His research interests include gas exchange and global change, digital image processing, and high-resolution infrared imagery.

Uwe Schimpf, Forschungsgruppe Bildverarbeitung, IWR
Universität Heidelberg, Im Neuenheimer Feld 368, D-69120 Heidelberg
uwe.schimpf@iwr.uni-heidelberg.de
http://klimt.iwr.uni-heidelberg.de

Johannes Schemmel is a PhD student at Heidelberg University, performing research on analog and mixed-signal VLSI systems for image processing. He studied physics in Heidelberg and received his diploma in physics from Heidelberg University in 1996.

Johannes Schemmel
IHEP, Universität Heidelberg
Schröderstrasse 90, 69120 Heidelberg, Germany
D-69120 Heidelberg, schemmel@asic.uni-heidelberg.de

Henning Scheich studied human medicine and philosophy from 1961 to 1966 in Colone, Munich, and Montpelier. Later he was assistant with Otto Creutzfeld at the MPI for psychiatry, Munich. From 1969 to 1972 he was assistant research neuroscientist at the University of California, San Diego. During the period from 1972 to 1974 he led a research group at the MPI for biophysical chemistry in Göttingen. From 1975 to 1994 he was professor at Darmstadt University of Technology. Since 1994 he has been director of the Leibniz Institute for Neurobiology in Magdeburg. His research includes acoustic control of behavior, learning and cognition, functional organization of the auditory cortex, speech processing, and imaging methods.

Henning Scheich, Leibniz Institute for Neurobiology
Brenneckestr. 6, 39118 Magdeburg, Germany
scheich@ifn-magdeburg.de, http://www.ifn-magdeburg.de

Steffen Schmalz received his diploma in computer science from University of Dresden in 1994. He was employed with Fraunhofer Gesellschaft from 1992 until 1994. He has been a member of the IMA research group at the department of computer science, University of Hamburg, since 1994. His research interests are in 2-D/3-D object recognition with active vision systems, machine learning, and fuzzy image processing.

Dipl.-Inform. Steffen Schmalz, Universität Hamburg
Fachbereich Informatik, AG IMA, Vogt-Kölln-Str. 30
D-22527 Hamburg, Germany, schmalz@informatik.uni-hamburg.de
http://ima-www.informatik.uni-hamburg.de/˜schmalz

Harald Schönfeld worked until December 1997 as a research assistant at the Chair for Optics. His main research topics are 2-D and 3-D image processing and sensor calibration.

Harald Schönfeld
Siemens AG, Medizinische Technik
Computertomographie CTE 2
Siemensstraße 1, D-91301 Forchheim, Germany
E-mail: hs@undine.physik.uni-erlangen.de

Thomas Scholz received his diploma and doctoral degree in physics from Heidelberg University in 1991 and 1995. He worked at the Interdisciplinary Center for Scientific Computing and the Institute of Environmental Physics on parallelizing of wavelet decompositions and depth-from-focus methods for industrial applications. Currently he is affiliated with SAP AG, Walldorf, Germany.

Dr. Thomas Scholz,
SAP-AG, Walldorf, Germany
Thomas.Scholz@sap-ag.de
http://www.sap-ag.de

Dominik Schmundt graduated in 1995 from the University of Heidelberg with a Master degree in physics. He is now pursuing his PhD at the Interdisciplinary Center for Scientific Computing, Heidelberg on quantitative analysis of leaf growth using image sequence processing.

Dominik Schmundt
Interdisciplinary Center for Scientific Computing
Universität Heidelberg, Im Neuenheimer Feld 360
D-69120 Heidelberg, Germany
Dominik.Schmundt@iwr.uni-heidelberg.de
http://klimt.iwr.uni-heidelberg.de/~dschmun/

Ulrich Schurr is research assistant at the Institute of Botany at the University of Heidelberg. He is a member of the research unit “Image Sequence Analysis to Study Dynamic Processes.” His main research interests are growth, gas exchange, and nutrient transport in plants.

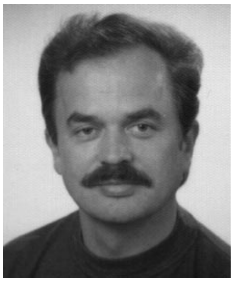
Ulrich Schurr, Institute of Botany
Universität Heidelberg, Im Neuenheimer Feld 360
D-69120 Heidelberg, Germany
uschurr@botanik1.bot.uni-heidelberg.de
http://klimt.iwr.uni-heidelberg.de/PublicFG/index.html

Klaus Singer worked several years as a freelancer in designing test and measurement equipment for psychophysiological research purposes before he joined a middle-sized German company in 1979 as a development engineer. In 1983 he became a shareholder and managing director of the company. He established its image processing department, was responsible for national and international marketing, and for key accounts in the field of image processing. Since 1997 he has been working as a consultant in industrial imaging. Since 1994, he has been a member of the directorate "Industrielle Bildverarbeitung/Machine Vision," Verein Deutscher Maschinen- und Anlagenbauer, Frankfurt (VDMA).

Klaus Singer, Consultant, F1 professional services GmbH
Ginsheimer Straße 1, 65462 Ginsheim-Gustavsburg, Germany
phone: +49 6134 55740-0, fax: 55740-3, kghsinger@t-online.de

Pierre Soille received the engineering degree from the Université catholique de Louvain, Belgium, in 1988. He gained the doctorate degree in 1992 at the same university and in collaboration with the Centre de Morphologie Mathématique of the Ecole des Mines de Paris. He then pursued research on image analysis at the CSIRO Mathematical and Information Sciences Division, Sydney, the Centre de Morphologie Mathématique of the Ecole des Mines de Paris, and the Abteilung Mustererkennung of the Fraunhofer-Institut IPK, Berlin. During the period 1995-1998 he was lecturer and research scientist at the Ecole des Mines d'Alès and EERIE, Nîmes, France. Now he is a senior research scientist at the Silsoe Research Institute, England. He worked on many applied projects, taught tutorials during international conferences, co-organized the second International Symposium on Mathematical Morphology, wrote and edited three books, and contributed to over 50 scientific publications.

Prof. Pierre Soille, Silsoe Research Institute, Wrest Park
Silsoe, Bedfordshire, MK45 4HS, United Kingdom
Pierre.Soille@bbsrc.ac.uk, http://www.bbsrc.ac.uk



Gerald Sommer received a diploma (1969) and PhD (1975) degree in physics from Friedrich-Schiller-Universität Jena, Germany, and a habilitation degree (Dr.-Ing. habil.) in engineering from Technical University Ilmenau, Germany, in 1988. He also graduated in computer science and medical physics. From 1969 to 1991 he worked at the University Jena in several departments. From 1991–1993 he was the head of the division for medical image processing at the Research Center for Environment and Health in Munich-Neuherberg. Since 1993 he has been professor for computer science at the University of Kiel, Germany. He is leading a research group on cognitive systems. His research interests include the design of behavior-based systems, signal theory and signal processing, neural computation for pattern recognition and robot control. He chaired

the Computer Analysis of Images and Patterns Conference (CAIP'97) and the Workshop on Algebraic Frames for the Perception-Action Cycle (AFPAC'97).
Gerald Sommer, Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität Kiel, Preusserstr. 1–9
D-24105 Kiel, Germany, gs@ks.informatik.uni-kiel.de
http://www.ks.informatik.uni-kiel.de/

Hagen Spies graduated in January 1998 from the University of Heidelberg with a master degree in physics. He also received an MS in computing and information technology from the University of Dundee, Scotland in 1995. In 1998/1999 he spent one year as a visiting scientist at the University of Western Ontario, Canada. Currently he works as a researcher at the Interdisciplinary Center for Scientific Computing at the University of Heidelberg. His interests concern the measurement of optical and range flow and their use in scientific applications.
Hagen Spies, Forschungsgruppe Bildverarbeitung, IWR
Universität Heidelberg, Im Neuenheimer Feld 368
D-69120 Heidelberg, Germany, Hagen.Spies@iwr.uni-heidelberg.de
http://klimt.iwr.uni-heidelberg.de/˜hspies

Norbert Stein received his Dipl.-Ing. degree in electrical engineering (control theory) from the Technical University at Darmstadt, Germany. In 1980 he started working on medical image analysis at Deutsche Klinik für Diagnostik, Wiesbaden. In 1984 he received his PhD from the Technical University at Darmstadt. He founded VITRONIC Bildverarbeitungssysteme GmbH in 1984. He is a member of the board of section "Robotics and Automation" and chairman of subsection "Machine Vision," in the German Association for Machinery and Plant Manufacturer (VDMA), and a member of the board of the German Association for Pattern Recognition (DAGM). He is the owner and managing director of VITRONIC GmbH.
Dr.-Ing. Norbert Stein, VITRONIC Dr.-Ing. Stein Bildverarbeitungssysteme GmbH, Hasengartenstrasse 14a, 65189 Wiesbaden, Germany, st@vitromic.de

Uwe Stilla received the B.Sc. degree in electronics engineering from Gesamthochschule of Paderborn, Germany, the M.Sc. degree in biomedical engineering and the PhD degree in electrical engineering, both from University of Karlsruhe, Germany. Since 1989 he has been a lecturer in the Department of Medical Informatics at Fachhochschule of Heilbronn and University of Heidelberg. He is currently a research scientist with FGAN research center, Ettlingen, Germany. His research interests include remote sensing, computer vision, and human vision.
Dr. Uwe Stilla, FGAN-Forschungsinstitut für Informationsverarbeitung und Mustererkennung, Eisenstockstr. 12, D-76275 Ettlingen, Germany
usti@gate.fim.fgan.de

Michael Stöhr studied physics at the Technical University of Darmstadt, Germany. He is currently pursuing his PhD thesis at the Interdisciplinary Center for Scientific Computing at Heidelberg University. His research topic is flow dynamics close air/water interface and his research interests include measurements of the flow field near the free water surface with the aid of 3-D particle tracking velocimetry and numerical simulation of flow applying numerical finite element solvers on Navier Stokes partial differential equations.

Michael Stöhr, Forschungsgruppe Bildverarbeitung, IWR
Universität Heidelberg, Im Neuenheimer Feld 368
D-69120 Heidelberg, Germany
Michael.Stoehr@iwr.uni-heidelberg.de
http://klimt.iwr.uni-heidelberg.de

Songyan Sun received the M. Sc. from Harbin Institute of Technology, Harbin, China, in 1983 and the PhD degree from the Technische Universität Clausthal, Clausthal-Zellerfeld, Germany, in 1992, both in electrical engineering. From 1983 to 1986, he worked at Harbin Institute of Technology, in stochastic and digital signal processing. He did research work from 1986 to 1988 at Universität Hannover in speech signal processing and recognition. Since 1992, he has been a RD engineer at Vitronic Bildverarbeitungssysteme GmbH, Wiesbaden, Germany. His current works are pattern recognition and neural networks, optical and handwritten character recognition, and real-time applications of image processing in the industry.

Songyan Sun, VITRONIC Dr.-Ing. Stein Bildverarbeitungssysteme GmbH
Hasengartenstr. 14a, D-65189 Wiesbaden, Germany
Phone: +49-611-7152-48, sun@vitronic.de

Ulrich Thönnessen received the B.Sc. degree in electronics engineering from Fachhochschule of Krefeld, Germany, and the M.Sc. degree in informatics from University of Aachen (RWTH), Germany. He is currently a research scientist with FGAN-Forschungsinstitut für Informationsverarbeitung und Mustererkennung, Ettlingen, Germany. His interests in computer vision are automatic recognition and classification of objects in image sequences, motion analysis, structural image analysis, and software and hardware for image processing.

Dipl.-Inf. Ulrich Thönnessen
FGAN-Forschungsinstitut für Informationsverarbeitung und Mustererkennung
Eisenstockstr. 12, D-76275 Ettlingen, Germany, thoe@gate.fim.fgan.de

Dietmar Uttenweiler is a research fellow at the II. Institute of Physiology at the University of Heidelberg in the group of Prof. Dr. R. H. A. Fink. He studied physics in Freiburg and Heidelberg. In 1990–1991 he worked at the University of Sussex, UK, supported by an Erasmus scholarship. He graduated as Diplom-Physiker in 1994 and received his doctoral degree (Dr. rer. nat.) in physics in 1997 from the University of Heidelberg. His research interests in biophysics comprise fluorescence imaging techniques, mathematical modeling, and digital image processing, in particular for the study of motor proteins and the calcium regulation of force generation in muscle.

Dr. Dietmar Uttenweiler, II. Physiologisches Institut
University of Heidelberg, Im Neuenheimer Feld 326, D-69120 Heidelberg
dietmar.uttenweiler@urz.uni-heidelberg.de

Thomas Vetter received his diploma in mathematics and the PhD degree in physics from the University of Ulm, Germany. He held postdoctoral positions at the Biophysical Department at the University of Ulm and at the Center for Biological and Computational Learning at the Massachusetts Institute of Technology. In 1993 he joined the Max-Planck-Institut für Biologische Kybernetik in Tübingen, Germany. His current research interests are image understanding and image synthesis.

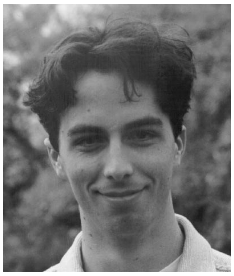
Thomas Vetter, Max-Planck-Institut für Biologische Kybernetik, Spemannstr. 38, 72076 Tübingen, Germany
thomas.vetter@tuebingen.mpg.de, http://www.tuebingen.mpg.de

Thomas Wagner received a diploma degree in physics in 1991 from the University of Erlangen, Germany. In 1995, he finished his PhD in computer science with an applied image processing topic at the Fraunhofer Institute for Integrated Circuits in Erlangen. Since 1992, Dr. Wagner has been working on industrial image processing problems at the Fraunhofer Institute, from 1994 to 1997 as group manager of the intelligent systems group. Projects in his research team belong to the fields of object recognition, surface inspection, and access control. In 1996, he received the "Hans-Zehetmair-Habilitationsförderpreis." He is now working on automatic solutions for the design of industrial image processing systems.

Dr.-Ing. Thomas Wagner, Fraunhofer Institut für Intregrierte Schaltungen
Am Weichselgarten 3, D-91058 Erlangen, Germany
wag@iis.fhg.de, http://www.iis.fhg.de

Rolf Watzel received his diploma in computer engineering from Darmstadt University of Technology in 1994. From 1994 to 1997 he was research assistant at the Federal Institute for Neurobiology in Magdeburg, Germany. Currently he is research assistant at the Department of Computer Engineering at Darmstadt University of Technology. He is working on his doctoral (PhD) dissertation.

Rolf Watzel, Fachbereich Elektrotechnik und Informationstechnik, Technische Hochschule Darmstadt
Merckstr. 25, D-64283 Darmstadt, Germany
row@dtro.tu-darmstadt.de
http://www.dtro.e-technik.tu-darmstadt.de/cgi-bin/row



Mark Oliver Wenig has studied physics in Münster and Heidelberg and received his diploma in 1998. He is now working on his PhD thesis on a scholarship from the Graduiertenkolleg “Modeling and scientific computing in mathematics and natural sciences” as a member of the research groups Digital Image Processing and Atmospheric Chemistry. His thesis is part of a project aiming at retrieving and analyzing atmospheric trace gas concentrations, a cooperation between the Interdisciplinary Center for Scientific Computing (IWR) and Institute of Environmental Physics (IUP), Heidelberg.

Mark O. Wenig, Institut für Umweltphysik
Im Neuenheimer Feld 366, D-69120 Heidelberg, Germany
mark.wenig@iwr.uni-heidelberg.de
http://klimt.iwr.uni-heidelberg.de/˜mwenig

Dieter Willersinn received his diploma in electrical engineering from Technical University Darmstadt in 1988. From 1988 to 1992 he was with Vitronic Image Processing Systems in Wiesbaden, working on industrial applications of robot vision and quality control. He then took a research position at the Technical University in Vienna, Austria, from which he received his PhD degree in 1995. In 1995, he joined the Fraunhofer Institute for Information and Data Processing (IITB) in Karlsruhe, where he initially worked on obstacle detection for driver assistance applications. Since 1997, Dr. Willersinn has been the head of the group, Assessment of Computer Vision Systems, Department for Recognition and Diagnosis Systems.

Dr. Dieter Willersinn, Fraunhofer Institut IITB, Fraunhoferstr. 1
D-76131 Karlsruhe, Germany, wil@iitb.fhg.de

Georg Wiora studied physics at the University of Karlsruhe from 1990 to 1997. Between 1995 and 1997 he worked on the development of a microscopic topographical measurement system at the Fraunhofer Institut für Chemische Technologie (ICT) in Pfinztal, Germany. Since 1997 he has been working on his doctorate at the DaimlerChrysler research center in Ulm (Germany) on calibration methods for 3-D fringe projection sensors and reliable optical measurement systems.

Georg Wiora, DaimlerChrysler AG
Research and Development
Wilhelm-Runge-Str. 11, D-89081 Ulm, Germany
georg.wiora@DaimlerChrysler.com

Hans-Joachim Wünsche studied electrical engineering at the Technische Universität, München, Germany from 1977–1980. From 1980–1982 he was a graduate student in Aerospace Engineering at the University of Texas, Austin, TX, on a Fulbright Scholarship, and completed his study with the M.S. in aerospace engineering. From 1982–1987 he was a research assistant with Prof. Dickmanns, ISF at the UniBwM. He received his PhD in engineering in 1987. He was technical manager at MBB Industrial Automation Technology (part of Daimler Benz Aerospace (DASA)) from 1987–1991 working on the development of autonomous guided vehicles for factories, robot systems, and laser welding systems and from 1991–1994 vice president Mechanical Division, Schroff GmbH. Since 1994 he has been president of Schroff UK Ltd., Hemel Hempstead, Great Britain. His research interests include motion detection and control of autonomous systems using dynamic machine vision.

Dr. Hans-Joachim Wünsche
Kings Langley, WD4 9NE, Great Britain

Christoph Zierl received his diploma in computer science from the Technische Universität, München, Germany in 1994. He is currently a PhD student at the Forschungsgruppe Bildverstehen FG BV (computer vision research group) at the same institution. His main research interests are CAD-based computer vision and vision for autonomous mobile robots, especially model-based object recognition and indexing techniques for the generation of object hypotheses.

Christoph Zierl, Forschungsgruppe Bildverstehen (FG BV)
Informatik IX, Technische Universität München
80290 München, Germany
Christoph.Zierl@in.tum.de
http://wwwradig.in.tum.de/people/zierl/

1 Introduction

Bernd Jähne

Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR)
Universität Heidelberg, Germany

Contents

The third volume of the Handbook on Computer Vision and Applications is centered on system aspects and applications. The system aspect actually has two meanings. First, there is the question of how all components that make up a computer vision system and that have been discussed in the first two volumes of the handbook are put together as a system in an optimal way. Second, in real applications, a visual task is only part of a larger system. Thus, there is the question of how it is integrated in the best way.

This introduction covers some basic questions of the architecture of computer vision systems in Section 1.1. Is it different from general-purpose computer systems? What are its basic function modules and what are the critical performance parameters?

Parts II and III of this volume are devoted to industrial, technical, and scientific applications of computer vision. This collection of application reports raises some important questions for theoreticians and practitioners alike. A theoretician may ask himself why his technique has not found its way into real-word applications. Or he may find out that his research tackles problems no one is interested in. Conversely, the practitioner may ask himself why he is still sticking with his inefficient and old-fashioned techniques when there are much better available.

This volume creates an opportunity for the necessary dialogue between basic research and applications. It is also the hope that the solutions presented in the application reports turn out to be useful in other application areas.

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

ISBN 0-12-379773-X/$30.00

1.1 Computer vision architecture

It is necessary to assume at the outset that no image acquisition or display is needed. Is there then any difference in the architecture of a computer vision system from a general computer? Yes? No? The answer is no, as a general-purpose computer is built only to be used to compute a wide range of tasks. It is thus only a question of how efficiently modern personal computer (PC) and workstation architectures can be used for computer vision tasks. The answer is yes, as for many applications it is required to put a stream of image data at a specified rate through the computer system. This requires so-called *real-time* computer hard- and software. Nowadays three trends are observable with respect to the platforms for computer vision applications.

Standard PCs. In recent years there has been a swift move away from dedicated—especially VME-bus based—computer system to standard PC-based systems for industrial applications. They have reached critical performance levels with respect to computing power and memory bandwidth for image processing (Section 3.2 and Chapter 11). This trend will only be accelerated in the near future by the addition of multimedia instruction sets, which make PCs more suitable for the processing of pixel data (Chapter 3).

Intelligent cameras. Rather simple tasks can be performed by so-called intelligent cameras. The camera includes the entire hardware to process the acquired images in a standalone system. Examples of applications with such systems are discussed in Chapter 13.

Configurable hardware. Field programmable gate arrays have acquired enough computing power to be suitable for sophisticated image processing (Chapter 2).

Computer architecture for computer vision systems would be incomplete if only hardware were to be considered. The best hardware is only as good as the software running on it. One may argue that the long tradition in imaging of dedicated hardware has considerably hindered progress in software. Portable, modular, and reusable software is especially critical for computer vision tasks because they tend to include many different modules. This is why three chapters in this volume deal with software engineering for computer vision (Chapters 4–6). In addition, a fourth chapter discusses application-oriented assessment of computer-vision algorithms (Chapter 7).

Although a PC with a frame grabber and a camera can be regarded as a simple computer vision system, it generally is more than that. This can be seen by comparing computer vision systems with human or other biological vision systems. While it makes not much sense to imitate a biological system with a technical system, it is very useful to compare them on the basis of function modules as shown in Table 1.1.

Table 1.1: *Function modules of human and machine vision*

Task	Human vision	Machine vision
Visualization	Passive, mainly by reflection of light from opaque surfaces	Passive and active (controlled illumination) using electromagnetic, particulate, and acoustic radiation (Volume 1, Chapters 3 and 6)
Image formation	Refractive optical system	Various systems (see Volume 1, Chapter 4)
Control of irradiance	Muscle-controlled pupil	Motorized apertures, filter wheels, tunable filters
Focusing	Muscle-controlled change of focal length	Autofocus systems based on various principles of distance measurements
Irradiance resolution	Logarithmic sensitivity	Linear sensitivity, quantization between 8- and 16-bits; logarithmic sensitivity, for example, HDRC-sensors (Volume 1, Chapter 8)
Tracking	Highly mobile eyeball	Scanner and robot-mounted cameras (Chapter 9)
Processing and analysis	Hierarchically organized massively parallel processing	Serial processing still dominant; parallel processing not in general use

From the table it is immediately clear that a camera and a computer are only two parts of many in a computer vision system. It is also obvious that computer vision systems have a much richer variety and more precise function modules with respect to visualization, image formation, control of irradiance, focusing, and tracking. Thus, a camera with a PC is a poor vision system from the system perspective. It has no way to control illumination, adapt to changes in the illumination, move around to explore its environment, track objects of interest or zoom into details of interest.

Control of illumination is the basis of many powerful technical techniques to retrieve the 3-D structure of objects in images. This includes shape from shading and photometric stereo techniques (Volume 2, Chapter 19), active illumination techniques (Volume 1, Chapters 18 and 20). The active vision paradigm (Chapter 9) emphasizes the importance of an active response of the vision system to the observed scene, for example, by tracking objects of interest. Even more, many visual tasks can only be solved if they are operated in a closed loop, that is, an action is observed and controlled by a vision system in a feedback loop. This is known as the perception-action-cycle (Chapter 10).

Table 1.2: *Classification of tasks for computer vision systems*

Task	References
	2-D geometry
Position	Chapters 31
Distance	Chapters 40
Size, area	Chapters Chapter 29
Form & shape	Volume 2, Chapter 21; Chapters 8, 12, 19, 41
	Radiometry-related
Reflectivity	Volume 2, Chapter 19
Color	Chapters 8, 27, 40
Temperature	Volume 2, Chapters 2 and 5; Chapters 35, 36
Fluorescence	Volume 1, Chapter 12; Chapters 30, 34, 39, 40, 41
	Spatial structure and texture
Edges & lines	Volume 2, Chapter 10; Chapters 8, 28
Autocorrelation function	
Local wave number; scale	Volume 2, Chapter 4; Chapters 8
Local orientation	Volume 2, Chapter 10; Chapters 8
Texture	Volume 2, Chapter 12; Chapters 13
	High-level tasks
Segmentation	Volume 2, Chapter 21; Chapters 12, 29, 31, 37
Object identification	Chapters 12, 13
Object classification	Chapters 8, 13, 37
Character recognition (OCR)	Chapters 14
Model- and knowledge-based recognition and retrieval	Chapters 13, 25, 29

1.2 Classes of tasks

Applications of computer vision can be found today in almost every technical and scientific area. Thus it is not very helpful to list applications according to their field. In order to transfer experience from one application to another it is most useful to specify the problems that have to be solved and to categorize them into different classes.

Table 1.3: *Classification of tasks for computer vision systems related to dynamical 3-D scences.*

Task	References
	3-D geometry
Depth, 3-D optical metrology	Volume 2, Chapters 17 and 18; Chapters 16–23
3-D object shape	Volume 1, Chapters 17–20; Volume 2, Chapter 19; Chapters 17, 18, 21
Distance & size	Chapters 16 and 40
Area	Chapter 41
Form	Chapters 39 and 41
	Motion
2-D motion field	Volume 2, Chapters 13 and 14; Chapters 31, 32, 33
3-D motion field	Chapter 31
Tracking	Chapters 15, 38, 28
	3-D modeling
3-D object recognition	Chapters 17, 20, 22, 23
3-D object synthesis	Chapters 20, 24
Tracking	Chapters 15 and 38
Autonomous motion; navigation	Chapters 9, 26, 27, 28

An attempt for such a classification is made in Tables 1.2 and 1.3. The first table categorizes the tasks with respect to 2-D imaging, while the second table extends the tasks to the analysis of dynamical 3-D scenes. The second column contains references to chapters dealing with the corresponding task. References to Volume 1 or Volume 2 generally describe the techniques themselves, while references to this volume describe applications.

Part I

Architecture of Computer Vision Systems

2 Field Programmable Gate Array Image Processing

Klaus-Henning Noffz[1], Ralf Lay[1], Reinhard Männer[2,3], and Bernd Jähne[3]

[1]Silicon Software GmbH, Mannheim, Germany
[2]Lehrstuhl für Informatik V, Universität Mannheim, Germany
[3]Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), Universität Heidelberg, Germany

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

2.1 Introduction

The high number of pixels in digital images work as a multiplication factor for most image processing algorithms. In particular, high-resolution 2-D images, 3-D images, or motion sequences require an enormous amount of computing power. Combined with real-time requirements, digital image processing puts hard constraints on the image processing hardware.

The big advantage in image processing, however, is the structure of the algorithms, which usually allows a high degree of parallelization. A second feature is the usually low number of bits required per pixel. Thus, digital image processing is greatly applicable to parallel processing by relatively simple units.

Today the trend in image processing is towards the use of general purpose hardware (Chapter 11). The architecture of microprocessors, however, is poorly suited to the structure of image processing algorithms. In a microprocessor instructions are processed sequentially and its architecture is optimized for complex operations on large operands. It is thus not surprising that both the emerging multimedia instructions sets such as *MMX* (Chapter 3), which in general purpose microcomputers exploit a certain level of parallelism, and the dedicated *Digital Signal Processors* (DSP) provide noticeable improvements over standard processors.

Superior performance is available using special purpose chips (*Application Specific Integrated Circuits*, ASIC). The ASICs are optimized for a certain image processing task, usually exploiting the full inherent parallelism. By adjusting the internal data path to the required number of bits their silicon is used very effectively. Typical fields of applications are graphic (video) cards, 3-D accelerators, or MPEG chips.

While ASICs provide enormous computing power they have some serious disadvantages. The biggest one is their lack of flexibility due to the restriction of a single functionality. Moreover, the development of an ASIC is an expensive and time-consuming enterprise. Upgrades always require chip redesign. Due to the development time and costs of ASICs, only a very limited set of the most important functions is usually implemented in special purpose chips. This is why we see ASICs only in mass market applications.

In this chapter we introduce the *FPGA coprocessor* as an alternative approach. This new class of computer combines the high speed of special purpose hardware with a flexibility that approaches that available with software solutions. The basic idea is to use freely programmable, highly complex logic ICs, *Field Programmable Gate Arrays* (FPGA) as computational core. These FPGAs consist of a high number of simple logic cells, which can be "wired" by loading a configuration program into the chip. An algorithm to be executed is realized as a special wiring, that is, as special purpose architecture on standard hardware. Nearly arbitrary applications can be programmed into and executed by the FPGA. Because the hardware itself is configured, such an implementation has the high computing power of ASICs. On the other hand, an *FPGA coprocessor* is reprogrammable within milliseconds. Just by pressing a button a completely new hardware configuration—which we call a *hardware applet*—is loaded that executes a totally different application. Thus *FPGA coprocessors* behave much like the usual computer.

This article begins with a brief introduction to the technology of FPGAs (Section 2.2.1) followed by the discussion of their computing power and expected technological advances (Section 2.2.2). Section 2.3 discusses the concept of the *FPGA coprocessor* that makes the advantages of FPGAs exploitable to image processing systems. After a general introduction in Section 2.3.1 an example *FPGA coprocessor* is discussed along with the commercial *FPGA coprocessor microEnable*[1]. The following Section 2.4 discusses the development of applications together with the optimization of image processing algorithms for *FPGA coprocessors* (Section 2.4.3) and the integration in general image processing software (Section 2.4.4). This chapter ends with the description of some example applications implemented on the system (Section 2.5).

2.2 Field programmable gate arrays (FPGAs)

2.2.1 Architectures of field programmable gate arrays

Field programmable gate arrays (FPGAs) are a relatively young technology introduced in 1984 by Xilinx. They consist of a 2-D matrix of simple logic cells (*Configurable Logic Blocks*, CLB), which can be arbitrarily connected by an interconnect network. Both the function of the cells and the “wiring” of the interconnections are freely programmable by downloading a configuration bitstream into the device.

The innovation introduced by the FPGAs was the indefinite number of reconfiguration cycles (each within a couple of milliseconds) and the high complexity far above existing programmable ICs. In terms of logical resources the largest FPGAs today compete with large Standard Cell ASICs and medium complexity Full Custom ASICs.

The architecture of an FPGA, shown in Fig. 2.1, consists of two elements, the functional plane and the programming plane. The programming plane is invisible for the user and implements physically the configuration of the FPGA. It consists of a high number of simple storage elements, each controlling the function of a certain part of the functional plane. During the configuration every element is loaded with one bit of the configuration bitstream. The by far most popular and

[1] *MicroEnable* is available from Silicon Software GmbH, Mannheim, Germany.

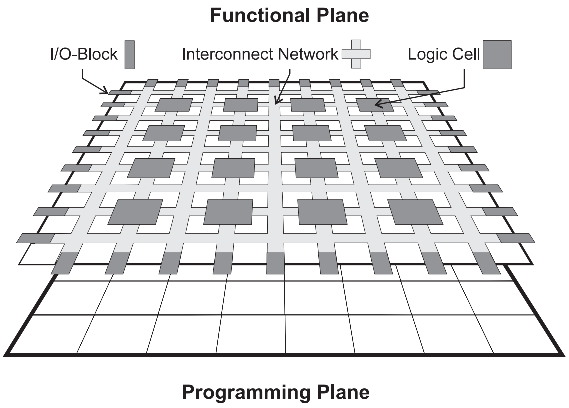


Figure 2.1: *The internal setup of a FPGA.*

for *FPGA coprocessors* most important implementation of the programming plane is SRAM cells connected to a very large shift register. This implementation provides the highest configuration speed and an indefinite number of configuration cycles. Alternative implementations are antifuses restricted to a single configuration and EEPROMs that allow about 1000 configuration cycles with strongly reduced speed.

In the user-visible functional plane the actual application is run. It basically consists of a 2-D regular matrix of programmable logic cells. The most widespread architecture of a logic cell is small Look-Up Tables (LUT) allowing the implementation of arbitrary Boolean functions of 4 to 6 input variables. Modern FPGAs provide several thousand logic cells.

To implement an algorithm on an FPGA the circuit representing the application is divided into small subcircuits fitting into these cells. The necessary connections among the logic cells are implemented by the interconnect network. It consists of a large number of wires and switch transistors that allow arbitrary connections to be programmed.

The third element of the functional plane is I/O cells surrounding the logic cell matrix. They connect the FPGA to the outside world. Each I/O cell is programmable as input, output, or bidirectional pin.

For a more detailed discussion on different types and technologies including quantitative analysis of FPGA architectures see Brown et al. [1]; for an overview of available FPGA chips and features see Oldfield and Dorf [2].

2.2.2 Computing power

In this section we discuss the computing power of FPGAs and what further advances in the FPGA technology are to be expected. In order

Table 2.1: *Estimated computing power of a Xilinx XC40125-0.9 FPGA for additions and multiplications*

Algorithm	Speed [MHz]	MIPS
Addition/Subtraction 8 bit	100	115,000
Multiplication 8 bit const	100	46,000
Multiplication 8 bit	100	30,000

to define the computing power of an ASIC one simply measures the throughput for the application for which the ASIC is built. For systems not restricted to a single task (like a CPU), the computing power is usually measured at maximum performance or, using benchmarks, under typical conditions. For an *FPGA coprocessor* the situation is in general the same as for a CPU. Due to programmability of the hardware itself, however, variations in performance are much higher. It depends strongly on both the instruction (from simple bit masking to a complex floating point division) and parameters such as the bit width used.

For image processing an FPGA implementation gains its performance by parallel or pipelined processing. The less complex the basic operation the more processing elements can be implemented in the FPGA and the higher the computing power (assuming an arbitrarily high degree of parallelism in the algorithm). Or vice versa: If a basic element requires a substantial part of the logical FPGA resources there will not be much gain from using FPGAs.

In Table 2.1 the computing power for a XC40125 FPGA—the largest available Xilinx FPGA—is shown assuming representative basic operations (add, sub, mult) for image processing. The resulting computing power of the FPGA is nearly two orders of magnitude higher than that of modern microprocessors such as a 400-MHz Pentium II processor (800 MIPS) or even high-end DSPs like the TMS320C60 (1200 MIPS). Even with MMX code with up to 6,400 million operations/s for 8-bit additions (Chapter 3) the FPGA gains more than one order of magnitude.

It is important to note that this performance estimation depends strongly on the basic operations. If, for example, the dominating instruction of the algorithm is less complex than an addition or multiplication, for example, an FPGA will allow an even higher speedup. For a rather complex basic instruction the gain drops rapidly. The conclusion from Table 2.1 is that FPGAs have the potential to gain an enormous speedup in image processing. They are best used in highly parallel algorithms implementable with comparatively simple basic elements. A good example is the field of real-time image preprocessing.

Table 2.1 reflects the situation today. How will the technology of FPGAs improve in time? Will a standard CPU be as fast as an FPGA

Table 2.2: *FPGA logic density trend for Xilinx XC4000 FPGAs*

Year	FPGA	CLBs	Factor	Comment
1993	4010	400	1.0	
1995	4025	1024	2.5	
1997	4085	3136	7.8	
1998	40125	4612	11.5	
1998	40250	9216	25.0	Announced

Table 2.3: *FPGA speed trend for Xilinx XC4000 FPGAs*

Year	FPGA	Clock rate [MHz]	Factor
1994	4000E-4	133	1.0
1996	4000E-2	192	1.4
1998	4000XL-1	400	3.0

some years from now? Or will the gap between FPGA and CPU become greater and greater?

The computing power of an FPGA is proportional to the product of the maximum frequency of the implemented circuit and the degree of parallelization. If we assume that problems in image processing are highly parallel we conclude that the computing power of an FPGA is proportional to the product of the number of primitive logic cells (the FPGA resources) and the speed of the internal logic (the maximum toggle frequency for the internal flipflops).

Internally FPGAs are basically SRAM—the higher the number of SRAM cells in the programming plane, the higher the number of primitive cells and the higher the FPGA complexity. Thus it is expected that the technological advances of FPGAs will follow the increases for SRAM in general (for both speed and storage size). An investigation of the growth of these two factors over the last three decades can be found in Vuillemin [3], in which the author predicts an increase in computing power for FPGAs by a factor of two each year!

In order to prove this conclusion, which is based on the general development of the SRAM technology, Tables 2.2 and 2.3 display the increase of FPGA speed and complexity during the last few years for an existing FPGA family—the Xilinx XC4000 FPGAs[2].

[2]The toggle frequency of the internal flipflops is calculated by the inverse of the sum of the clock-to-output time and flipflop setup time neglecting internal routing. All numbers are from Xilinx data books from the year given.

For the last few years both speed and complexity increased by almost 40 % (speed) to 60 % (complexity) per year. This corresponds to an annual speedup in computing power of over a factor of two per year, confirming the prediction of Vuillemin [3].

The speedup in computing power for CPUs, however, has also followed an exponential growth track over the last 40 years. According to Thacker [4], the growth kept constant for more than 30 yr by an annual increase factor of 1.25—thus the computing power doubles every 3 yr.

If these trends are stable, the performance of FPGA will increase many times faster than CPUs. The reason for this behavior is that the computing power of CPUs benefits mainly from advances in process technology by the increase of the CPU's maximum frequency. Architectural improvements usually have a much lower impact. By contrast, FPGAs take very direct advantage of both effects because the higher number of logic cells is directly used for a higher degree of parallelism.

Due to their advantage for bit-level algorithms and integer arithmetic we expect that *FPGA coprocessors* will play an important role in image processing. Probably even algorithms requiring floating point operations will become more and more applicable to *FPGA coprocessors*.

2.3 FPGA-based image processing systems

In 1989, five years after the advent of FPGAs, the first FPGA-based computing machines appeared that made use of the computing power and flexibility of this new technology. The basic idea is to establish a new class of computer consisting of an array of FPGAs as computing core. As in a conventional von Neumann computer, arbitrary applications are run by downloading an appropriate configuration bitstream (the software).

The first of these *FPGA processors* were large systems optimized for maximum computing power [5, 6]. They consist of dozens of FPGAs, reaching or partially outrunning the computing power of supercomputers. Important examples were a DNA sequencing application outperfoming a CRAY-II by a factor of 325 [7] or the world's fastest RSA implementation [8]. A wide variety of tasks were implemented on some of the machines [9]. The large number of image processing applications demonstrated the potential of *FPGA processors* for this area [10]. A good collection of pointers to most *FPGA processors* can be found in Guiccone [11].

Due to their high speed, large *FPGA processors* typically run their application independently of their host computer—the host only provides support functions. Typical image processing systems, however, usually need tight integration of data aquisition, image processing, and

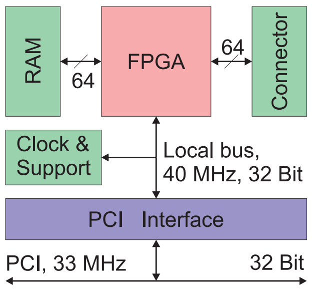


***Figure 2.2:** Functional elements of microEnable.*

display routines. This makes a certain subclass of *FPGA processors*—*FPGA coprocessors*—great candidates for image processing.

An *FPGA coprocessor* is closely coupled to a conventional computer and optimized for fast communication with the CPU. It works similarly to the old mathematical coprocessors but executes in contrast to them complete algorithms instead of single instructions. The *FPGA coprocessors* usually contain only a few or one FPGA on relatively small boards. The following description of the hard- and software architecture is made using the example of the *microEnable* coprocessor[3]. It is particularly suitable to image processing applications.

2.3.1 General architecture of FPGA coprocessors

An *FPGA coprocessor* generally consists of three components (Fig. 2.2):

1. *The FPGA* is the computing kernel in which algorithms are executed.
2. *The memory system* is used as a temporary buffer or for table lookup.
3. *The I/O interface* is required for communication between host and the FPGA coprocessor. On some machines it also comprises a second interface to external electronics.

The overall performance of the FPGA coprocessor depends on the performance of all three subsystems: FPGA; memory system; and I/O interface.

2.3.2 Example system: the microEnable

Like *FPGA coprocessors* in general, the hardware of *microEnable* (Fig. 2.3) comprises the three functional units FPGA, memory system, and I/O interface. The I/O interface of the processor additionally provides a secondary port to external electronics. Used as interface to image sources,

[3] *MicroEnable* is a commercially available system provided by Silicon Software GmbH, Mannheim, Germany.

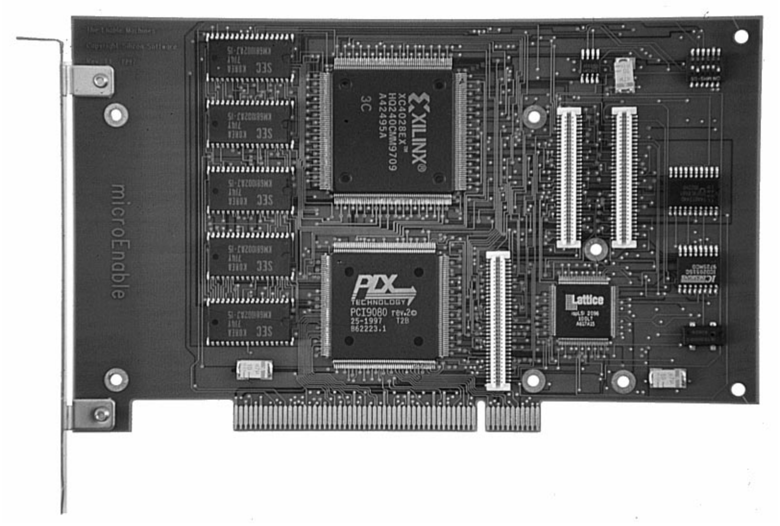


Figure 2.3: *The microEnable board.*

this feature is important for image processing making *microEnable* an intelligent framegrabber. The last functional unit is the clock and support circuitry of the processor. In the next paragraphs the hardware setup is described in more detail.

The *FPGA* is *microEnable*'s computing kernel. Due to the fast growth of FPGA complexity, and in order to keep the system inexpensive and simple, the board contains only a single device of the Xilinx XC4000 family. It supports all XC4000E/EX devices larger than the XC4013. Since 1998, a 3.3 V version is available covering all XC4000XL FPGAs between XC4013XL and XC4085XL. This makes the computing power of the processor scalable by one order of magnitude.

The *RAM system* is a fast buffer exclusively used by the FPGA. It consists of 0.5 to 2 MBytes of fast SRAM. This buffer is intended for tasks such as temporary data buffering, table lookup, or coefficient storage. Experiences in different fields of applications have shown the importance of a fast RAM system.

The *PCI interface*—the physical connection to the host computer and thus to the application—supports the standard PCI bus of 32-bit width and 33 MHz. It is implemented in a custom PCI chip, the PCI9080 from PLX Technology. The advantages of this implementation are the high performance of the chip, a wide variety of features, and the excellent DMA support important for high transfer rates. In our experience these features compensate for the drawbacks compared to an FPGA implementation: the need for a PLX interface on the local bus side and to cope with a (related to the number of features) sophisticated device.

For user-specific functionality and communication with external electronics a daughterboard can be plugged onto *microEnable*. Two types of *daughterboards* are supported:

1. *Common Mezzanine Cards (CMC)* following the IEEE P1396 specification; and
2. *S-LINK cards* following a specification developed at the European particle physics lab CERN [12].

The *CMC* is a mechanical specification defining the board mechanics, connector type, and pinout for some signals (clock, reset, power, ground) of the daughtercard [13]. In addition to the CMC specification, other specifications define the protocol used: the most important one is the PMC specification for the PCI protocol. *MicroEnable* supports the 33 MHz, 32-bit PMC specification. The CMC/PMC daughterboards are most widely used by FPGA coprocessors [14, 15].

The *clock and support circuitry* of *microEnable* implements tasks such as configuration and readback of the FPGA, a JTAG port, and the provision of user-programmable clocks. For applications running on *microEnable*'s FPGA four different clocks are available. The main clocks are two phase synchronous clocks freely programmable in the range between 1 and 120 MHz (at 1-MHz steps). The ratio between these two clocks can be set to 1, 2, or 4. The third clock is the local bus clock programmable in the interval between 1 and 40 MHz. The last clock signal usable is an external clock signal provided by the S-LINK connector.

2.3.3 Device driver and application interface library

A high bandwidth bus like PCI does not necessarily guarantee a high data exchange rate between FPGA coprocessor and CPU. What ratio of the pure hardware performance is usable in reality by the application depends strongly on the device driver. The *microEnable* provides device drivers for the operating systems *Linux* and *WindowsNT 4.0* and thus supports the most important aspects of PCs. The device drivers support master as well as slave accesses, two independent gather/scatter DMA channels, and interrupt capability (generated by either DMA or the FPGA). (For a sample interface, see Fig. 2.4.)

In order to provide users with a simple but fast programming interface, accesses to *microEnable* are memory mapped into user space. To the user, these accesses appear as usual memory accesses.

Although the memory map is a fast mechanism, the reachable data bandwidth is far below the theoretical limit of 132 MBytes/s. The key issue for maximum throughput is DMA transfers. Using DMA *microEnable* provides a performance of up to 125 MBytes/s, 95 % of the peak

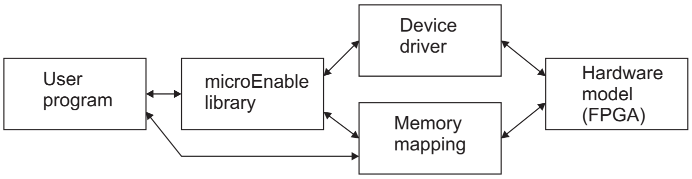


Figure 2.4: *The software interface to microEnable.*

PCI performance, measured to/from a user application running in user space under WinNT 4.0 on a 233 MHz Pentium II.

The DMA transfers data in large blocks. In many real-life applications the hardware applet (Section 2.3.4) is not able to accept data in each cycle (this is usually data dependent). The dilemma here is that a normal DMA transfer would overrun the hardware applet and cause data loss. To do it without DMA would decrease performance unacceptably. *MicroEnable* allows DMA transfers using a handshake protocol with the applet running in the FPGA. Using this mechanism transfer rates comparable to normal DMA rates are possible.

High performance is only one important factor; the other one is to provide the user a simple view to the *FPGA coprocessors*' functionality. This depends on the device driver and application interface library. *MicroEnable* provides a C library for user applications. The complete setup of the board is hidden behind a few rather simple function calls:

Example 2.1: Code example for board setup

```
main() {
   microenable my_processor;
   fpga_design my_design;
   unsigned long *access;
   unsigned long value;

   // initialize microEnable
   initialize_microenable(&my_processor);

   // load the hardware applet into memory
   load_design(&my_processor, &my_design,"design_file.hap");

   // now configure the FPGA
   configure_fpga(&my_processor);

   // memory map on microEnable
   access = GetAccessPointer(&my_processor);

   // access the FPGA
   access[0] = 100;        // write
   value    = access[0];    // read
}
```

Table 2.4: *Configuration times for* microEnable

FPGA	Time in ms
4013E	30
4028EX	90
4036EX	105

The distinction between `load_design()` storing the configuration bitstream into memory and `configure_fpga()` performing the actual configuration allows an effective exchange of FPGA designs during processing. At the beginning of the application a few configuration bitstreams are stored using `load_design()`. During the execution of the actual application the FPGA can be quickly reprogrammed using `configure_fpga()`.

2.3.4 Hardware applets

A certain application running on the FPGA coprocessor is called a *hardware applet*. The term not only refers to the FPGA configuration but also comprises the software routines that allow the host software to communicate with the coprocessor, a data basis for information like the supported FPGA type, the maximum clock frequency, the available address space, or the register (and their parameters implemented in the FPGA). The user does not need additional information to run the applet.

Hardware applets can be loaded in a very short time depending on the FPGA size (Table 2.4).

2.3.5 Modes of operation

MicroEnable can be run in three different modes of operation that are of particular interest for image processing applications.

1. *Virtual mode.* The short reconfiguration times of *microEnable* allow dynamic switching between different hardware applets during the run of an application. A complex image processing task is divided into several subtasks implemented in an applet each. By dynamically exchanging the applets the *FPGA coprocessor* behaves like a much larger FPGA: it becomes “virtual hardware.” This mode enhances the capabilities of the *FPGA coprocessor* and saves hardware resource costs. The virtual mode also can be combined with the two modes that follow.

2. *The offline hardware accelerator mode*. In the offline mode *microEnable* accelerates image processing algorithms on image data already available in the PC's main memory.
3. *The online hardware accelerator and data grabber mode*. In the online mode *microEnable* has direct access to the external image source. The source, usually a digital camera, is directly connected to *microEnable* via a daughterboard. By appropriate programming of the FPGA the processor acts like a conventional framegrabber. Because the framegrabbing requires only a fraction of the FPGA resources the processor additionally executes image processing as in the offline mode. *MicroEnable* then combines the functionality of a conventional framegrabber with real-time image processing capabilities.

Because in the offline mode the image data are transferred twice over the PCI bus, there is a potential bottleneck—in some algorithms RAM system and FPGA are able to process the image at a higher rate than that with which PCI can cope. This problem can be solved using the online mode.

2.4 Programming software for FPGA image processing

The higher the FPGA complexity the more important are fast, reliable, and effective programming tools. State-of-the-art are those hardware description languages (HDL) that have been used in chip design for years. The most common languages are *VHDL* and Verilog. Currently, *microEnable* supports *VHDL*—the Verilog interface that was announced in 1998.

For FPGA programming *FPGA coprocessors* introduce a new challenge due to their tight coupling to a CPU. Usually there is much interaction between CPU and an *FPGA coprocessor*—the extreme is real hardware/software codesign where the application is divided into a part running on the *FPGA coprocessor* and one on the microprocessor. In order to simulate this behavior correctly, a hardware/software cosimulator is required. Because this feature is not supported by current HDL tools a programming language called *CHDL* (*C++ based hardware description language*) was developed.

A programmer using an HDL describes the architecture of a circuit representing the application. Compared to the algorithmic description used in common programming languages for software the level of abstraction is lower, thereby leading to a longer development time for a given algorithm. Although there is an increasing effort to develop high-level languages for *FPGA coprocessors* there is no commonly accepted language today.

The following sections describe the *VHDL* and *CHDL* programming tools for *microEnable*. High-level language support is expected by the end of 1998.

2.4.1 VHDL

The *VHDL* support of *microEnable* is a large library to be used with any commercial *VHDL* compiler. A major part of this library includes I/O modules like master and slave interfaces, DMA channels, and DMA channels using a handshake protocol. For each of these modules there exists a corresponding C routine from the Application Interface Library. A programmer instantiating a DMA handshake module in his *VHDL* code uses a corresponding DMA routine in his application software. There is no need to deal with *microEnable*'s address space nor with the timing and function of the local bus. This mechanism simplifies the integration of hardware applets into the application software.

Another important class of modules are interfaces to *microEnable*'s RAM system and the external connectors. There is no need for programmers to care about pinout or timing of the external RAM or interfaces. The *VHDL* library can be used with arbitrary *VHDL* synthesis and simulation tools.

Compared to schematic entry, *VHDL* has advantages that relate to its higher level of abstraction and optimal code reuse. Among the drawbacks are the difficulty in generating optimized units exploiting special features of the FPGA logic cells (usually this is only possible by compiler/vendor specific constraints) and the (usual) lack of a C interface for simulation. While the first argument only applies to a limited class of low-level library elements, the missing C interface is an important drawback during verification of complex algorithms. The big problem here is the verification of the hardware/software codesign—the interaction between the part running on the hardware and the application software. To express this interaction in a *VHDL* testbench is a time-consuming and hazardous enterprise.

2.4.2 CHDL

To solve the hardware/software cosimulation problem a new language called *CHDL* (*C++ based hardware description language*) was developed. The syntax of *CHDL* is similar to common HDLs like *ABEL* or *Altera HDL*. Anyone familiar with such languages will learn it in a couple of hours. *CHDL* is implemented in C++ classes. Only an ordinary C++ compiler is required to run it.

Compared to *VHDL* the important advantage is testbench generation. The basic concept behind *microEnable*'s *CHDL* simulator is that there is no difference between the testbench generation/verification and the final application software (i. e., the part controlling the hardware). The whole testbench is written as a normal C++ program accessing *microEnable*'s internal address space, initiating DMA transfers, or generating interrupts—exactly the way the application software communicates with *microEnable*. The important difference is that all accesses are redirected from the hardware to the simulator. Each access defined in the testbench software will be translated into a clock cycle-based animation of the defined *CHDL* program. Using this mechanism, it is straightforward to transfer whole images (this may cover some 100,000 clock ticks) via a virtual DMA to the simulator, and to read the processed image back into a buffer via another virtual DMA.

In order to verify the *CHDL* code the user can now compare this buffer with expected results or simply display the resulting image on screen. It is also possible to watch selected signals defined in *CHDL* on a waveform display. For algorithms using *microEnable*'s external RAM the simulator provides routines to virtually access the current content of this memory.

This mechanism has three major advantages:

- The whole testbench generation and verification is programmed in usual C++, which makes the simulation process fast and convenient.
- Even complex interactions between host software and the hardware applet are easily simulated. This feature is very important for hardware/software codesign, when applications are divided between the host CPU and the FPGA hardware. This type of simulation is extremely difficult for *VHDL* testbenches where only the content of a (C generated) test vector file is usually loaded into the simulator.
- The testbench generator software is identical to the software required to control the FPGA coprocessor during the run of the real application. The simulator thus guarantees the correspondence of the application control software with the simulated circuitry. For complex hardware/software codesign applications the interaction between coprocessor and software is not trivial.

These advantages speed up the development of applets strongly and make simulation results more reliable.

In addition to the easy testvector generation *CHDL* provides the user with the full C++ functionality. Instead of defining single modules it is straightforward to declare whole classes of modules: Libraries thus appear more like library generators.

Module respectively class definitions in *CHDL* are hierarchical to any level. The language allows direct access to all special features of FPGA cells including placement and routing constraints. This eases the generation of highly optimized modules.

The disadvantage, however, is the lower level of abstraction compared to *VHDL* (e.g., there is no behavioral description). This drawback is evident in practice mostly when state machines are defined. For this reason a state machine generator was developed that allows state machines to be defined using a common `switch case` syntax.

The migration between *VHDL* and *CHDL* is simple. Code written in *CHDL* can be directly exported to *VHDL* (for both synthesis and simulation).

CHDL is available for the operating systems Linux, Solaris, and WinNT 4.0.

2.4.3 Optimization of algorithms for FPGAs

FPGAs offer the opportunity to design a hardware that is optimally adapted to a specific problem. Thus the optimization of a given algorithm for FPGAs has many more degrees of freedom than on a standard microprocessor. Some of these possibilities are briefly outlined.

Multiple processing units. For each task it can be decided how many processing units run in *parallel* and/or which topology of a pipeline is the best choice. This includes the choice between a few rather complex processing units and a network of simpler units.

Variable bit-length processing. On a standard microprocessor the data types are fixed to multiple bytes. The integer arithmetic/logical unit and multiplication units have a width of 32 bits or wider — much wider than the data from imaging sensors. This waste of resources can be avoided with FPGAs. The bit lengths of any processing unit can be set no wider than the required depth. Thus it is possible to balance speed versus accuracy as it is best suited for a given task.

Special processing units. A standard microprocessor knows only a few standard types of processing units. For integer processing this includes an arithmetic/logical unit, a shifter, and a multiplication unit. With FPGAs any type of special processing unit that might be required for a specific task can be implemented. Such dedicated processing units have a higher performance and/or consume fewer resources than standard processing elements.

Lookup table processing. While multimedia instruction sets have considerably boosted the performance of standard microprocessors, the computation of histograms and lookup table operations cannot be accelerated by such instructions (see Chapter 3). These FPGA processors do not show this deficit. It is very easy to implement lookup table operations. They can be used for different operations such as multiplication with a fixed constant, computing the square, square root or any other nonlinear function.

2.4.4 Integration into a general software system

If an FPG processor is not used as a standalone system, it must be integrated into a general software environment that coordinates the processing on the FPGA with the host system. Basically, two concepts are possible.

On the one hand, an FPGA image processing system that receives data from an image source can be handled as a freely programmable frame grabber (online hardware accelerator, Section 2.3.5). The FPGA processor receives the raw image data, processes them and transfers the processed data to the host. Thus the data flow is basically unidirectional (except for downloading hardware applets to the FPGA during initialization, Section 2.3.4). The host just controls the processing mode, starts and stops the data flow, and directs the incoming data for further processing.

On the other hand, one or multiple FPGAs could be used as coprocessors in a host system. In this case, the flow of data is much more complex. The host sends the data to be processed to the FPGA processors and receives the processed data. This bidirectional data transfer is much more difficult to handle—especially if several FPGA processors run in parallel—and requires much more input/output bandwidth.

Thus the concept of the freely programmable frame grabber is by far preferable. It is the method of choice provided that the FPGA processor is powerful enough to process the incoming image data in real time. This approach fits very well into the modern concepts for frame buffers. Typically the software interface consists of two parts, a configuration utility and a device driver. The configuration utility is typically an interactive software module that allows the user to configure the frame grabber for the application. The user sets the camera to be used, the image resolution, the trigger signals, etc. and saves these settings in a configuration file. This configuration file is then loaded during initialization of the frame grabber and configures the frame grabber to the preselected values. This concept is ideally suited to choose hardware applets for FPGA processors (Section 2.3.4). At runtime one of the processing modes built into the hardware applet including a pass-through mode transferring the original image data to the host for direct data control is selected. The mode and frequency with which the image data are acquired and processed are controlled via standard routines to grab and snap images as they are available in any frame grabber software interface. Of course, it should also be possible to load a new applet at runtime for a different type of application.

With this concept it is also very easy to perform parallel processing on the FPGA and the host. The software on the host just needs a number of synchronization routines to perform further processing on the host in sync with the incoming preprocessed image data. In addition, it is necessary to direct the incoming preprocessed data into image objects of the image processing software running on the host.

Such features are, for instance, built into the heurisko® image processing system.[4] In this software, objects can be allocated in such a way that background transfer via DMA over the PCI bus is possible. Such buffers can be individual images or image sequences that can be used as image ring buffers. Background acquisition into such an object is started by the `AcqStart()` instruction and stopped by the `AcqStop()` instruction. With the `AcqCnt()` instruction it can be inquired how many frames have been transferred since the last call to `AcqStart()`. `AcqTest()` inquires whether currently a processed frame is transferred and `AcqWait(n)` waits until the next n frames are acquired. With these commands it is easy to synchronize further processing with the incoming preprocessed image data. As long as the capacity of the PCI bus is not exceeded, the data streams from multiple FPGA processors can be processed in this way.

2.5 Application examples

Since the first FPGA processors began to be used, many applications were implemented, including many image processing algorithms. We will concentrate here on several industrial applications in the field of image processing implemented on *microEnable* during 1997 and 1998. Exemplary applications, which can be used for online or offline analysis (see the different operating modes of *microEnable* described in Section 2.3.5), are image preprocessing, filtering, and JPEG compression.

2.5.1 Image preprocessing

Image preprocessing generally covers simple steps that process the acquired image data for the actual image analysis. Usually these preprocessing steps have to be performed directly after the image acquisition. For this reason it is often important to execute image preprocessing concurrently with the acquisition. Examples of image preprocessing steps are:

1. Cut out of regions of interests.
2. Rotation by 0°, 90°, 180°, 270°.

[4]heurisko has been developed by AEON, Hanau, Germany in cooperation with the University of Heidelberg (http://www.aeon.de).

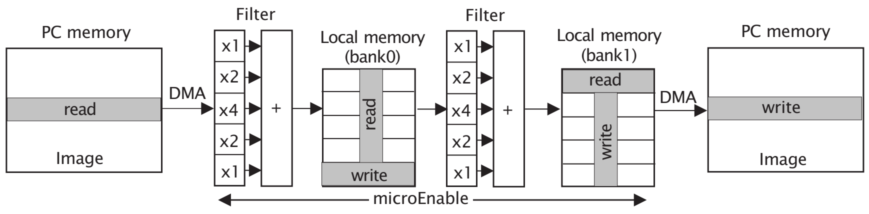


***Figure 2.5:** Data flow for 2-D convolution with a 5×5 binomial filter.*

3. Scaling.
4. Reformatting, for example, the creation of BMP compatible pictures.
5. Color space conversion, for example, from RGB to YUV space or vice versa.

The examples listed here were implemented on *microEnable* in the scope of an industrial application. All subalgorithms were implemented in a single hardware applet. With this standard configuration the functionality of an average digital frame grabber is covered by the *FPGA processor*.

However, the use of an *FPGA processor*, especially its reprogammability, offers some additional features. One is the possibility to extend the described standard hardware configuration with user specific applications. This allows us to solve additionally user-specific image processing steps on-the-fly. The user gains the high speed-up by executing the application directly in hardware. Example applications are unusual data formats or on-the-fly convolution for data filtering. Due to their algorithmic structure these highly parallel or bit-shuffling algorithms often can not be executed by software with sufficient speed. On the other hand, these algorithms are too problem-specific for an ASIC solution. The second feature is dynamic reconfiguration of the hardware, that is, to load different hardware configurations sequentially. Instead of a single one the user has several hardware setups at his or her disposal.

2.5.2 Convolution

An important operation in digital image processing is *convolution*. In what follows, we consider an implementation of a binomial filter for *microEnable. Binomial filters* , typically used as smoothing filters (Volume 2, Chapter 7), belong to the class of *separable filters*—thus the convolution can be separated in horizontal and vertical direction (Volume 2, Section 5.6).

The implementation of this filter consists of two concurrent processing steps (compare Volume 2, Section 5.6.2, especially Fig. 5.7):

1. The first unit is the filter pipeline in a horizontal direction. The data are processed line-by-line as they are shifted through the filter pipeline. The lines convolved by the horizontal filter mask are stored in bank 0 of the local memory (Fig. 2.5).
2. Concurrently to the horizontal convolution, the vertical convolution takes place in the second filter stage. A vertical stripe equivalent to the width of the vertical mask is read out from the intermediate storage area in bank 0 and processed by a second, identical filter pipeline. The results are stored in bank 1 of *microEnable* 's local memory (Fig. 2.5) and—in a consecutive step—readout per DMA to the host again row by row.

This type of filter is excellently qualified for an implementation in FPGAs. As already mentioned, the actual filter is implemented in a pipeline, which can be built of many stages (ordinal of 8 or higher without any problem). The execution speed is nearly independent of the order of the filter operation. If the coefficients are constant, they are implemented directly in hardware, if not, as loadable registers.

The implementation in *microEnable* achieves 50 MBytes/s throughput with a 9×9 separable filter kernel, limited by the PCI bus bandwidth. If the filter is used in an online mode avoiding the PCI bottleneck, 100 MBytes/s throughput is achieved. With 18 multiplications and 16 additions per pixel this corresponds to a sustained computing power of 3400 MOPS. This it just about the peak performance of a 400 MHz Pentium II with MMX instructions (Section 3.4.2, Table 3.4). The real performance on this platform is, however, only about 300 MOPS, 10 times slower than the FPGA (Section 3.6, Table 3.6).

To implement several filters serially, a ping pong mode is possible, where data is transferred and processed from one memory bank to the other and vice versa. Exploiting the possibility of reconfiguration, the hardware can be adapted to any arbitrary filter combination. Because of the parallel implementation, the execution time for two sequential separable filters in *microEnable* is the same as for a single one (however the number of required resources is higher).

2.5.3 JPEG compression

The JPEG is the most popular lossy compression standard for still images [16]. The data flow of the JPEG compression algorithm is shown in Fig. 2.6. It is based on a *discrete cosine transform* (DCT) of the image (to reduce the 8×8 pixels). The JPEG exploits the fact that the DCT coefficients representing high image frequencies can be suppressed with minor effects to the overall image quality. This is done in the second processing step called *quantization*. The computed DCT coefficients are divided and rounded to integer numbers, in which the divisor is

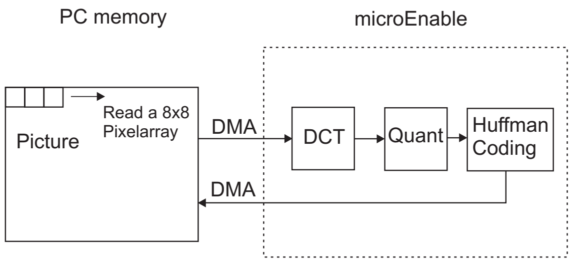


Figure 2.6: *The data flow of the JPEG compression.*

larger for higher DCT frequencies. Usually this step results in long sequences of zero values. The quantized values are now compressed using traditional compression methods like *Huffman coding*.

The complexity of the algorithm is dominated by the DCT requiring 16 concurrent MACs (multiply and accumulate) per pixel[5]. Due to the nature of the quantization step there is no need for floating point arithmetic.

The JPEG implementation on *microEnable* reaches an overall data throughput of 18.3 MBytes/s running at 30 MHz on an XC4036EX. This covers processing time, data transfer to and from memory, and additional tasks like rotation and format conversions. The implementation also fits into an XC4028EX but is limited to 9.8 MBytes/s. Programming was done in *VHDL* with a development time of 12 weeks.

2.6 Conclusions

Today's fast growth of FPGAs in both speed and complexity makes FPGA coprocessors a very interesting solution for image processing. In suitable applications *microEnable* is able to outperform traditional CPUs by at least one order of magnitude. The announced one million gate FPGAs will make this advantage even higher.

The problem of highly complex FPGA coprocessors is more on the side of programming languages. To develop applications for such large devices in an acceptable amount of time, high-level languages are urgently needed. This is underscored by our experience with a quite sophisticated algorithm like JPEG.

Acknowledgments

The authors gratefully acknowledge the financial support of the Wirtschaftsministerium of the state Baden-Württemberg within the Open-Eye project.

[5]Depending on the algorithm used this number can be reduced to 11 multiplications. If a scaled output is sufficient it can be further reduced to 5 multiplications.

2.7 References

[1] Brown, S. D., Francis, R. J., Rose, J., and Vranesic, Z. G., (1992). *Field-Programmable Gate Arrays*. Boston: Kluwer Academic Publishers.

[2] Oldfield, J. and Dorf, R., (1995). *Field-Programmable Gate Arrays*. New York: Wiley-Interscience.

[3] Vuillemin, J. E., (1994). On Computing Power. In *Programming Languages and System Architectures*, J. Gutknecht, ed. Stuttgart: Springer.

[4] Thacker, C. P., (1992). *Computing in 2001*. Technical Report, Digital Equipment Corporation, Systems Research Center, 130 Lytton, Palo Alto, CA94301.

[5] Bertin, P., Roncin, D., and Vuillemin, J., (1989). Introduction to programmable active memories. *Systolic Array Processors*, pp. 300–309.

[6] Gokhale, M., Holmes, W., Kopser, A., Kunze, D., Lopresti, D., Lucas, S., Minnich, R., and Olsen, P., (1990). SPLASH. A reconfigurable linear logic array. In *Proceedings of the International Conference on Parallel Processing*.

[7] Lopresti, D. P., (1991). Rapid implementation of a genetic sequence comparator using field-programmable gate arrays. In *Advanced Research in VLSI, Proceedings of the 1991 University of California/Santa Cruz Conference*, C. H. Séquin, ed., pp. 138–152. Cambridge, MA: MIT Press.

[8] Shand, M. and Vuillemin, J., (1993). Fast implementation of RSA cryptography. In *Proceedings of the 11th IEEE Symposium on Computer Arithmetic, Windsor, ONT, Canada*, pp. 252–259.

[9] Bertin, P., Roncin, D., and Vuillemin, J., (1993). Programmable active memories: a performance assessment. In *Research on Integrated Systems—Proceedings of the 1993 Symposium on Research on Integrated Systems*, G. Borriello and C. Ebeling, eds., pp. 88–102. Cambridge, MA: MIT Press.

[10] Athanas, P. M. and Abbott, A. L., (1994). Processing images in real time on a custom computing platform. In *Field-Programmable Logic: Architectures, Synthesis, and Applications*, R. W. Hartenstein and M. Z. Servít, eds., pp. 156–157. Berlin: Springer.

[11] Guiccone, S., (1998). List of FPGA-based Computing Machines. http://www.io.com/˜guccione/HW_list.html.

[12] Boyle, O., McLaren, R., and van der Bij, E. *The S-LINK Interface Specification*. ECP Division, CERN, Technical Report edition, (1996). Available under http://www.cern.ch/HSI/s-link.

[13] IEEE Standards Department. *Draft Standard for a Common Mezzanine Card Family: CMC*, P1396/Draft 2.0 edition, (1995).

[14] Digital Equipment Corporation, (1997). PCI Pamette V1. http://www.research.digital.com/SRC/pamette.

[15] Virtual Computer Corporation, (1997). The H.O.T. Works PCI Board. http://www.vcc.com.

[16] W. B. Pennebaker, J. M., (1993). *JPEG Still Image Data Compression Standard*. New York: Van Nostrand Reinhold.

3 Multimedia Architectures

Bernd Jähne[1] and Helmut Herrmann[2]

[1]Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR)
Universität Heidelberg, Germany
[2]AEON Verlag & Studio, Hanau, Germany

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

3.1 Introduction

From the machine and computer vision that began in the late 1970s, a rich tradition of hardware-oriented approaches to vision problems has evolved (Section 11.2) because general purpose hardware was simply not powerful enough for the demands of real-world computer vision tasks. This approach also had serious deficiencies and turned out to be a significant obstacle to more rapid progress.

First, it takes considerable effort to implement a certain algorithm on a dedicated hardware platform. The developers have to acquire knowledge of the platform and the development tools before they can write even a single line of code. Moreover, the development environment is normally much less comfortable and user friendly than on standard platforms. Secondly, a specific hardware platform has quite a short lifetime and may be of use only for a certain class of applications. Thus, a large fraction of development time is wasted just in reimplementing code from one platform to another. The opportunities for reusing code are minimal. The constant pressure to adapt algorithms to new hardware hinders a quick transfer of new and more powerful algorithms from research to applications. Moreover, developers are reluctant in implementing complex algorithms because it takes too much effort.

The swift change in the machine vision industry away from dedicated industrial computer architectures based on VMEbus systems to standard PC-systems (Section 11.5) seems to indicate that standard hardware has become powerful enough to handle real-world vision tasks. The advantages of using standard hardware for computer vision tasks are obvious. The hardware is cheap and powerful and user-friendly development tools are available, which means that portable, modular and reusable software development is possible.

Thus it is timely to perform a systematic analysis of standard hardware platforms for computer vision tasks. We will focus on the recent development that integrates multimedia instruction sets into mainstream processors. What is the impact of this development on signal processing and computer vision? We will analyze all major instruction sets including Sun's *visual instruction set* (VIS), Intel's *multimedia instruction set* (MMX), AMD's *3DNow*, Digital's *MVI*, and Motorola's *AltiVec*.

In Section 3.2 a quantitative approach is taken to evaluate standard computer architectures for signal processing by analyzing the computing performance and data transfer capabilities. Section 3.3 introduces the SIMD-principle for pixel processing implemented with all of the multimedia instruction sets, and Section 3.4 gives a detailed comparative analysis of the instruction sets. Finally, Section 3.5 discusses how efficiently various basic signal processing algorithms can be implemented with SIMD architectures.

Table 3.1: *Peak performance of floating-point arithmetic of the Intel microprocessor family (after [1, 2]); the number of clocks refers to the throughput; if the duration (latency) is different from this figure, the latency of the operation is added in parenthesis*

Year	Processor	Clock rate [MHz]	Add [Clocks]	Add [MFLOPS]	Mul [Clocks]	Mul [MFLOPS]
1978	8087	4.7	70-100	0.055	130-145	0.034
1982	80287	12	70-100	0.141	90-145	0.102
1985	80387	33	23-34	1.16	25-37	1.06
1989	80486	100	8-20	7.1	16	6.3
1993	Pentium	166	1(3)	167	2(3)	83
1997	Pentium MMX	233	1(3)	233	2(3)	116
1998	Pentium II	400	1(3)	400*	2(5)*	200*

* separate execution units for addition and multiplication

3.2 Signal processing performance of microprocessors

Since its invention the development of the microprocessor has seen an enormous and still continuing increase in performance. A single number such as the number of *millions of operations per second* (MOPS) or millions of *floating-point operations per second* (MFLOPS) or the mere clock rate of the processor does not tell much about the actual performance of a processor. We take a more detailed approach by analyzing the peak computing performance, the bus data transfer rates, and the performance for typical signal processing tasks.

3.2.1 Peak computing performance

In evaluating the computing performance of a microprocessor the *peak performance* for an operation has to be strictly distinguished from the real performance for certain applications. The values for the peak performance assume that the data to be processed are already contained in the corresponding registers (or primary cache). Thus, the whole chain of transfer from the main memory to the registers is ignored (Section 3.2.2). The peak performance value also assumes that all instructions are already in the primary program cache and that they are set up in an optimum way for the specific operation. Thus, it is a rather hypothetical value, which will certainly never be exceeded. When one keeps in mind that real applications will miss this upper theoretical peak performance limit by a significant factor, it is still a useful value in comparing the performance of microprocessors.

Table 3.2: *Peak performance of integer arithmetic of the Intel microprocessor family (after [1, 2]); the number of clocks refers to the throughput; if the duration (latency) is different from this figure, the latency is added in parenthesis; note that the values for the processors with MMX instructions do* not *refer to these instructions but to the standard integer instructions*

Year	Processor	Clock [MHz]	Add [Clocks]	Add [MOPS]	Mul [Clocks]	Mul [MOPS]
1978	8086	4.7	3	1.57	113-118	0.04
1982	80286	12	2	6	21	0.57
1985	80386	33	2	16.7	9-22	2.13
1989	80486	100	1	100	13	7.7
1993	Pentium	166	1/2*	333	10	16.7
1997	Pentium MMX	233	1/2*	466	10	23.3
1998	Pentium II	400	1/2*	800	1 (4)	400

* 2 Pipelines, 2 ALUs

The peak performance of microprocessors can easily be extracted from the technical documentation of the manufacturers. Table 3.1 shows the peak performance for floating-point arithmetic including addition and multiplication of the *Intel microprocessor family*. From the early 8087 floating-point coprocessor to the Pentium II, the peak floating-point performance increased almost by a factor of 10,000. This enormous increase in performance is about equally due to the 100-fold increase in the processor clock rate and the reduction in the required number of clocks to about 1/100.

Significantly lower is the performance increase in integer arithmetic (Table 3.2). This is especially true for addition and other simple arithmetic and logical operations including subtraction and shifting because these operations required only a few clocks on early microprocessors. The same is true for integer multiplication up to the Pentium generation. The number of clocks just dropped from about 100 down to 10. With the Pentium II, the throughput of integer multiplications is 1 per clock, effectively dropping down the duration to 1 clock.

Thus, low-level signal processing operations that are almost exclusively performed in integer arithmetic do not share in the performance increase to the extent that typical scientific and technical number crunching applications do. In fact, the much more complex floating-point operations can be performed faster than the integer operations on a Pentium processor. This trend is also typical for all modern *RISC processors*. Although it is quite convenient to perform even low-level signal and image processing tasks in floating-point arithmetic, a 32-bit

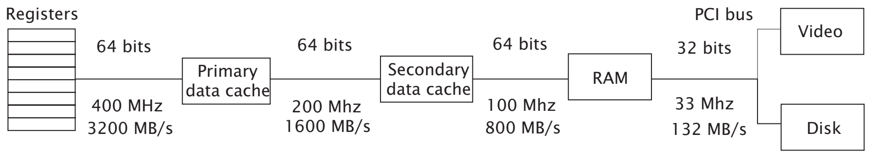


Figure 3.1: *Hierarchical organization of memory transfer between peripheral storage devices or data sources and processor registers (Pentium II, 400 MHz).*

floating-point image requires, respectively, 4 and 2 times more space than an 8- or 16-bit image.

3.2.2 Data transfer

Theoretical peak performances are almost never achieved in signal processing because a huge amount of data has to be moved from the main memory or an external image data source such as a video camera to the processor before it can be processed (Fig. 3.1).

Example 3.1: Image addition

As a simple example take the addition of two images. Per pixel only a single addition must be performed. Beforehand, two load instructions are required to move the two pixels into the processor registers. The load instructions require two further operations to increase the address pointer to the next operands. After the addition, the sum has to be stored in the destination image. Again, an additional operation is required to increase the address to the next destination pixel. Thus, in total six additional operations are required to perform a single addition operation. Moreover, because the images often occupy more space than the secondary cache of a typical microprocessor board, the data have to be loaded from the slow main memory.

One of the basic problems with modern microprocessors is related to the fact that the processor clock speed has increased 100-fold, while memory bus speed has increased only 10-fold. If we further consider that two execution units typically operate in parallel and take account of an increase in the bus width from 16 bits to 64 bits, the data transfer rates from main memory lack a factor of five behind the processing speed as compared to early microprocessors. This is why a hierarchical cache organization is of so much importance. Caching, however, fails if only a few operations are performed with a large amount of data, which means that most of the time is wasted for data transfer.

Example 3.2: Image copying

Copying of images is a good test operation to measure the performance of memory transfer. Experiments on various PC machines revealed that the transfer rates depend on the data and the cache size.

Copying data between images fitting into the second level cache can be done with about 180 MBytes/s on a 266-MHz Pentium II. With large images the rate drops to 80 MBytes/s.

A further serious handicap of PC image processing was the slow transfer of image data from frame grabber boards over the AT bus system. Fortunately, this bottleneck ceased to exist after the introduction of the *PCI bus*. In its current implementation with 32-bit width and a clock rate of 33 MHz, it has a peak transfer rate of 132 MB/s. This peak transfer rate is well above transfer rates required for real-time video image transfer. Gray-scale and true color RGB video images require transfer rates of 6–10 MB/s and 20–30 MB/s, respectively. With well-designed PCI interfaces on the frame grabber board and the PC mother board sustained transfer rates of up to 80–100 MB/s have been reported. This transfer rate is comparable to transfer rates from the main memory to the processor.

Standard personal computers have reached the critical threshold for real-time image data transfer. Thus, it appears to be only a question of time before standard digital camera interfaces are introduced to personal computers (Section 11.6.2).

3.2.3 Performance of low-level signal processing

The peak computing performance of a processor is a theoretical figure that cannot be reached by applications coded with high-level programming languages. Extensive tests with the image processing software heurisko showed that with optimized C programming the following real performances can be reached: 20-30 MFLOPS on a 166-MHz Pentium; 50-70 MFLOPS on a 200-MHz Pentium Pro; and 70-90 MFLOPS on a 266-MHz Pentium II. The integer performance is even lower, especially multiplication which took 10 clocks on a Pentium and more on older processors. This is why only half of the floating-point performance could be achieved. Further tests revealed that with assembler optimized coding of the inner loops of vector functions only modest accelerations would be gained. Usually, the speed-up factors are smaller than 2. These performances do not allow sophisticated real-time image processing with rates of 6-10 Pixels/s.

3.3 Principles of SIMD signal processing

3.3.1 History

The discussion at the end of the previous section clearly showed that standard microprocessors still do not have sufficient computing power for real-time low-level image processing tasks. If we analyze the microprocessor architecture we find two basic problems that limit the performance of such algorithms:

- Processing units that are built to perform computations with 32- to 80-bit numbers are used to perform computations with 8- to 16-bit image data; and
- 64-bit buses transfer only 8- to 16-bit data.

This severe mismatch of the standard microprocessor architecture for signal processing was recognized early and has also led to the development of dedicated *digital signal processors* (DSP). Such processors still play a significant role in highspeed and real-time signal and image processing.

There is, however, also a long-standing trend in including digital signal processing as an integrated part into standard microprocessors. By the end of the 1970s, Texas Instruments had built a freely programmable graphics processor, the TMS34010. This processor had 32-bit registers that could process pixels of 1 to 32 bits [3]. The trick was simply to pack as many pixels as could fit into one register and to process them in parallel. With special graphics instructions it was possible to perform simple arithmetic and logical operations between two pixels and to address them either linearly or directly in an xy addressing mode. Unfortunately, it was not possible to perform multiplication and shift operations. Thus, the instruction set was not suitable for even simple image processing algorithms such as convolutions.

The concept of performing the same operation with multiple pixels in a single register is a type of parallel processing known as *single instruction, multiple data* processing (SIMD). Like the TMS34010, the Intel i860 RISC processor also included SIMD pixel instructions that were only suitable for simple graphics operations (Z buffering, Gouraud shading) but not for signal processing. The first full-fletched SIMD pixel processing engine was developed by Motorola for the 88110 RISC processor [4]. Unfortunately, before this processor could be used in systems, the PowerPC had already succeeded it.

3.3.2 Multimedia instruction sets

A basic problem with these early designs was that there was no mainstream application that justified its development. Also, memory capacities and prices were still too high to allow mainstream applications to use visual data. This all changed with the multimedia wave and the advent of inexpensive high-capacity memory chips. This time the workstation manufacturer Sun was first. They included the *visual instruction set* (VIS) in the UltraSPARC architecture [5, 6]. Shortly after, Intel announced in spring 1996 the *multimedia instruction set* (MMX) [2, 7, 8]. In the second half of 1996 prototypes were available for developers and

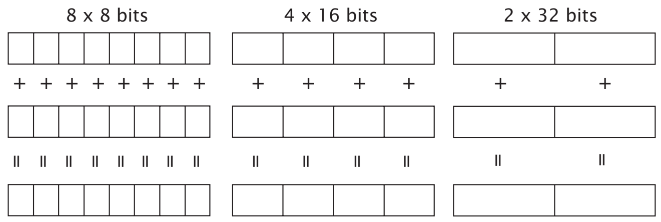


Figure 3.2: *Data types for multimedia instruction sets.*

in January 1997 the P55C was the first *Pentium processor* with the MMX instruction set.

Meanwhile, multimedia instruction sets have become a mainstream development area for microprocessor architecture. The *Pentium II* is the second Intel processor to include the MMX instruction set. The AMD included the MMX into their K6 processor [9] and introduced in May 1998 the K6-2 with an extension of the MMX instruction set called 3-DNow! that extends the SIMD principle to 32-bit floating-point arithmetics [10, 11]. Likewise, Cyrix included MMX in the MII processor and has plans similar to those of AMD to extend the MMX instruction set. It is expected that Intel will also soon come out with an extension of MMX.

Similar trends can be observed for other architectures. DEC plans to include a *motion video instruction set* (MVI) into the Alpha chip. In June 1998, Motorola announced the *AltiVec* instruction set for the PowerPC processor family [12].

3.4 Comparative analysis of instruction sets

In this section a detailed comparison of the available multimedia instruction sets is performed. The comparison includes Sun's VIS, Intel's MMX, AMD's 3-DNow!, and Motorola's AltiVec. DEC's motion video instruction set is not further considered because it is a highly dedicated instruction set with only a few instructions to compute the sum of absolute differences, compute minima and maxima, and pack/unpack pixels specifically targeted to improve motion video compression [13].

3.4.1 Registers and data types

All multimedia instruction sets except for Motorola's AltiVec work with 64-bit registers (Table 3.3). The AltiVec instruction set uses 32 128-bit registers. Because multiple pixels are packed into the multimedia registers a number of new data types were defined. The following data formats are supported by all instruction sets: packed bytes (8 bits), packed

Table 3.3: *Comparison of the register sets for several multimedia instruction sets*

Feature	MMX	3-DNow!	VIS	AltiVec
Number of registers	8	8	32	32
Width (bits)	64	64	64	128
Number of packed bytes	8	8	8	16
Number of packed words	4	4	4	8
Number of packed doublewords	2	2	2	4
Number of packed floating-point doublewords	-	2	-	4

words (16 bits), packed doublewords (32 bits) (Table 3.3, Fig. 3.2). Packed 32-bit floating-point doublewords are supported by the 3-DNow! and AltiVec instruction sets. The processors of the Intel family have only 8 multimedia registers, while the RISC architectures from Sun and Motorola are much more flexible with 32 registers [5].

The Intel MMX registers are mapped onto the floating-point register set. This approach has the advantage that no changes to the architecture were applied that would have caused software changes in operating systems for exception handling. The disadvantage is that floating-point and MMX instructions cannot be carried out simultaneously. With the extension of the MMX instruction set to include packed 32-bit floating-point arithmetic—as with AMD's 3-DNow! instructions and probably also with Intel's MMX2—this disadvantage is disappearing.

The 64-bit multimedia registers double the width of the standard 32-bit registers of 32-bit microprocessors. Thus twice the number of bits can be processed in these registers in parallel. Moreover, these registers match the 64-bit buses for transfer to the main memory and thus memory throughput is also improved.

3.4.2 Instruction format and processing units

The structure of all the various multimedia instructions reflects the basic architecture into which they are implemented. Thus the MMX instructions (and extensions of it) are two-operand instructions that are all of the same form except for the transfer instructions:

$$\mathrm{mmxreg1} = op(\mathrm{mmxreg1}, \mathrm{mmxreg2}|\mathrm{mem64}) \tag{3.1}$$

Destination and first operand is an MMX register; the first operand is overwritten by the result of the operation. This scheme has the significant disadvantage that additional register copy instructions are required if the contents of the first source register has to be used

Table 3.4: *Comparison of the peak performance of various implementations of multimedia instruction sets for 16-bit integer arithmetics. The three given numbers are: number of processing units, number of operations performed in parallel, number of clocks needed per operation. If the duration (latency) of the operation is different from the latter, the latency of the operation is added in parenthesis. The values for additions are also valid for any other simple arithmetic and logical instruction.*

Year	Processor	Clock [MHz]	Add	Add [MOPS]	Mul	Mul [MOPS]
1997	Pentium MMX	233	$2 \cdot 4 \cdot 1$	1864	$1 \cdot 4 \cdot 1(3)$	932
1998	AMD K6	300	$1 \cdot 4 \cdot 1$	1200	$1 \cdot 4 \cdot 1$	1200
1998	Pentium II	400	$2 \cdot 4 \cdot 1$	3200	$1 \cdot 4 \cdot 1(3)$	1600
1998	AMD K6-2	333	$2 \cdot 4 \cdot 1$	2667	$1 \cdot 4 \cdot 1(2)$	1333

more than once. The second operand can either be an MMX register or an address to a 64-bit source. Thus—in contrast to classical RISC architectures—it is not required to load the second operand into an MMX register before it can be used. One exception to the rule mentioned here are shift operations. In this case the shift factor can be given as a direct operand.

The visual instruction set of Sun's UltraSPARC architecture is a classical three-register implementation. Most instructions have one destination and two source registers:

$$\text{visreg1} = op(\text{visreg2}, \text{visreg3}) \tag{3.2}$$

This has the advantage that no source register is overwritten saving register copy instructions.

The AltiVec instruction set announced in June, 1998 is even more flexible. It allows instructions with one destination and up to three source registers.

The Intel P55C processor has a duration for all MMX instructions of just one clock Intel [2]. All arithmetic, logic and shift operations also have a latency of only one clock. This means that the results are immediately available for the instructions executed in the next clock cycle. Only the multiplication instructions show a latency of 3 clocks.

Both the Pentium P55C and the Pentium II have the following MMX processing units: two MMX ALUs, one MMX shift and pack unit, and one MMX multiplication unit. Because the MMX processors have two execution pipelines and four MMX execution units (two ALUs, a multiplier, and a shift unit) chances are good that two MMX instructions can be scheduled at a time. For simple instructions such as addition, this results in a peak performance of 1864 and 3200 MOPS on a 233-MHz MMX Pentium and a 400-MHz Pentium II, respectively (Table 3.4).

The performance figures for the AMD processors are quite similar. Although the K6 has only one MMX processing unit and thus can only schedule one MMX instruction at a time, the multiplication instructions do not show any additional latency. For integer operations, the AMD K6-2 is quite similar to the MMX Pentium with the exception that the multiplication instructions have a latency of only two clock cycles (Table 3.4).

The *AMD K6-2 processor* initiates the second wave of multimedia instruction sets extending SIMD processing to 32-bit floating-point numbers [10]. Two of them are packed into a 64-bit register. This parallel processing of only 32-bit floating-point figures still requires much less hardware than a traditional floating-point processing unit that processes either 80-bit or 64-bit floating-point numbers. Moreover, it can share much of the circuits for 32-bit integer multiplication. This extension brings 32-bit floating-point arithmetic to a new level of performance. With a throughput of 1 operation per clock cycle and the parallel execution of addition/subtraction and multiplication operations, the peak floating point performance is boosted to 1333 MOPS, way above the peak performance of 200 and 400 MFLOPS for floating-point multiplication and addition of a 400-MHz Pentium II.

No implementation of the AltiVec instruction set is yet available, so no detailed consideration can be made yet. Because of the 128-bit registers, however, it is evident that this architecture is inherently two times more powerful at the same processor clock speed and number of processing units.

3.4.3 Basic arithmetic

Integer arithmetic instructions include addition, subtraction, and multiplication (Table 3.5, Fig. 3.3). Addition and subtraction are generally implemented for all data types of the corresponding instruction set. These operations are implemented in three modes. In the wraparound mode arithmetic over- or underflow is not detected and the result is computed modulo the word length.

This behavior is often not adequate for image data because, for example, a bright pixel suddenly becomes a dark one if an overflow occurs. Thus a *saturation arithmetic* is also implemented often for both signed and unsigned values. This means that if an over-/underflow occurs with an operation, the result is replaced by the maximum/minimum value.

Multiplication is implemented for fewer data types than addition and subtraction operations and also not in saturation arithmetics (Table 3.5). In the MMX instruction set multiplication is implemented only

Table 3.5: *Basic arithmetic multimedia instructions*

Instruction set	8 bits	16 bits	32 bits	32 bits float
	Addition/Subtraction			
Sun VIS	W, US, SS	W, US, SS	W	-
Intel MMX	W, US, SS	W, US, SS	W	-
AMD MMX & 3-DNow!	W, US, SS	W, US, SS	W	Y
Motorola AltiVec	W, US, SS	W, US, SS	W	Y
	Multiplication			
Sun VIS	W (8×16)	W	-	-
Intel MMX	-	W	-	-
AMD MMX & 3-DNow!	-	W	-	Y
Motorola AltiVec	W, US, SS	W, US, SS	-	Y
	Multiply and Add			
Intel MMX	-	Y	-	-
AMD MMX & 3-DNow!	-	Y	-	-
Motorola AltiVec	W, US, SS	W, US, SS	-	Y

The abbreviations have the following meaning: US unsigned saturation arithmetic; SS signed saturation arithmetic; W wraparound (modulo) arithmetic; Y yes.

for packed 16-bit pixels in three variants. The first two store only either the 16 high-order or low-order bits of the 32-bit multiplication result. The third variant adds the 32-bit multiplication results pairwise to store two 32-bit results in two doublewords.

3.4.4 Shift operations

Shift operations are available with packed data types in the very same way as for standard integers. Thus, logical (unsigned) and arithmetic (signed) shifts with sign extension are distinguished. With the MMX instruction set shift operations are not implemented for packed bytes but only for packed words and doublewords and for the whole 64-bit word. In this way packed data can be shifted to other positions within the register. Such operations are required when data not aligned on 64-bit boundaries are to be addressed.

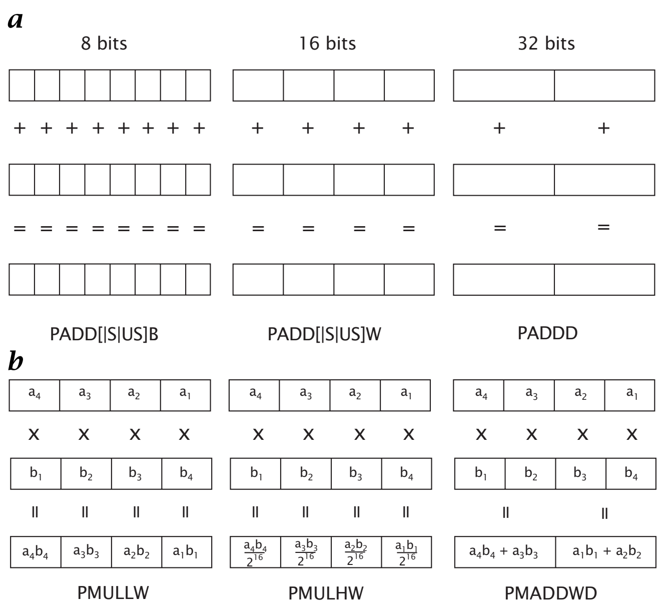


***Figure 3.3:** MMX **a** addition and **b** multiplication instructions.*

3.4.5 Logical operations

Logical operations work bitwise so it is not required to distinguish any different packed data types. Logical operations simply process all bits of the register. The MMX instruction set includes the following logical instructions: and, or, and exclusive or xor. In addition, a special and operation is available where the first operand is negated before the and operation. Motorola's AltiVec instruction set includes also a nor instruction and Sun's VIS knows in total 8 types of logical instructions.

3.4.6 Comparison, minimum, and maximum operations

Comparison operations are normally used to set flags for conditional branches. The comparison operations in multimedia instruction sets are used in a different way. If the result is true, all bits of the corresponding element that have been compared are set to one. If the result is false, all bits are set to zero. In this way a mask is generated that can be used for subsequent logical operations to select values on conditions and to compute minima and maxima without any conditional jumps that would stall the execution pipeline.

The MMX instruction set includes only greater and equal comparison operations as does Motorola's AltiVec instruction set. While the MMX instructions are only implemented for signed packed 8-, 16-, and 32-

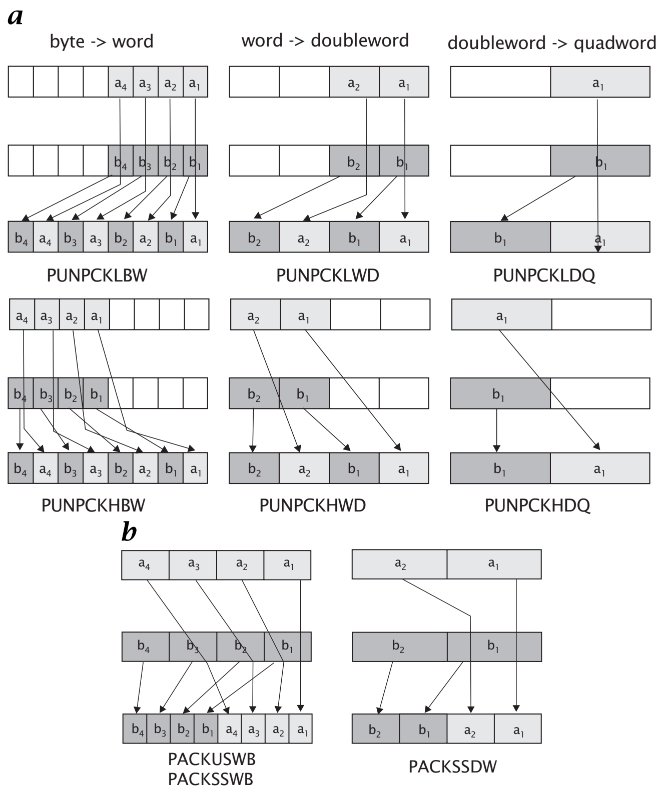


Figure 3.4: *MMX **a** unpack and **b** pack instructions.*

bit data, the AltiVec instructions are implemented for both signed and unsigned data types. Sun's VIS instruction set is much more flexible. All standard comparison instructions including greater than, greater equal, equal, not equal, less equal, and less than are implemented.

The MMX does not include any direct instructions to compute the minimum or maximum of two packed data words. Such instructions have been added for packed floating-point data with the 3-DNow! instruction extension. The AltiVec instruction set incorporates minimum and maximum instructions for both packed signed and unsigned integers and for packed floating-point data.

3.4.7 Conversion, packing, and unpacking operations

Conversion instructions convert the different packed data types into each other. Such operations are required because arithmetic operations must often be performed with data types wider than the width of the input data. The multiplication of two 8-bit values, for instance, results in a 16-bit product. Operations that increase the width of packed data types are called *unpack operations* (Fig. 3.4a). They are implemented as versatile merge operations by interleaving the packed data in the lower-order or higher-order word of two multimedia registers in the destination register.

Instructions that decrease the width are known as *pack instructions* (Fig. 3.4b). They take the packed data from two multimedia registers and truncate the bitlength to the half. The MMX instruction set includes only pack instructions with signed and unsigned arithmetic. The AltiVec instruction set also has modulo pack instructions.

3.4.8 Transfer operations

The load and store instructions of standard instruction sets work only with the standard bitlength of the integer registers. Therefore these instructions cannot be used to load and store the wider multimedia registers. With the MMX instruction set 32 and 64 bits, that is, a double or a quadword can be moved from and to memory. The quadword move instructions are the most efficient instructions because they utilize the full bus width with a single instruction.

These wide move instructions cause, however, an alignment problem. If an address is not aligned on a 64-bit boundary, which is normally the case with 8-, 16-, and 32-bit data types, memory access is slowed down. In order to take full advantage of the MMX instructions, it is necessary to align the data correctly. This requires a number of careful considerations detailed in the corresponding Intel manuals [2].

3.5 SIMD algorithms for signal processing

In this section we will discuss which classes of signal processing algorithms can be performed and how efficiently this can be done. This includes the following classes of operations: point operations, global transforms, convolution, gray-scale morphology, segmentation, binary morphology, classification, and neural networks.

3.5.1 Point operations

Any type of *point operation* can efficiently be performed with SIMD algorithms because all pixels can be processed in parallel. It is just required to perform a scan that includes all points of a D-dimensional signal and to perform the operation point by point.

Example 3.3: Image accumulation

The following code fragment shows the inner loop of a point operation that adds an 8-bit image to a 16-bit image in Intel MMX inline assembly code.

```
        pxor    mm7, mm7  // Set register mm7 to zero

vsbadd1:
// Load 4 x 4 pixels from 8-bit source into the
// low-order doubleword of registers mm0 - mm3
        movd        mm0, [esi]
        movd        mm1, [esi+4]
        movd        mm2, [esi+8]
        movd        mm3, [esi+12]
// Unpack from 8 bits to 16 bits, add to destination
        punpcklbw   mm0, mm7
        paddw       mm0, [edi]
        punpcklbw   mm1, mm7
        paddw       mm1, [edi+8]
        punpcklbw   mm2, mm7
        paddw       mm2, [edi+16]
        punpcklbw   mm3, mm7
        paddw       mm3, [edi+24]
// Save in destination
        movq        [edi], mm0
        movq        [edi+8], mm1
        movq        [edi+16], mm2
        movq        [edi+24], mm3
// Increment addresses and check loop counter
        add     esi, 16
        add     edi, 32
        sub     ecx, 1
        jg      vsbadd1
```

This loop contains 20 instructions that add per loop scan 16 pixels of the 8-bit source image to 16 pixels of the 16-bit destination image. As some instructions run in parallel (e. g., `punpcklbw` and `paddw`), the loop should take efficiently about one clock per pixel. On a 400-MHz Pentium II, it should thus run with a rate of 400-MPixels/s. This performance could, however, be achieved only if all source and destination data were available in the primary cache. Because this is never possible with image data, the performance of such a simple operation is rather limited to the maximum possible sustained memory transfer from and to the main memory. If we assume that the effective memory transfer rate is in the order of 100 MB/s and count only the load operations (3 bytes/pixel), the performance is slowed down by one order of magnitude to about 30 MPixels/s.

The preceding example shows that simple operations are memory-transfer limited. In other words, many more operations per pixel can be performed before the computing power of the multimedia instruction set limits the throughput.

There are two classes of operations that are related to point operations but cannot be accelerated by SIMD instructions: *lookup table* operations and the computation of *histograms*. Common to both operations is an additional indirection. The value of a variable is used to compute the address of the lookup table or the address of the element in the histogram that is to be incremented. Because of the content-dependent addresses, these operations cannot be performed with SIMD instructions.

This is a serious limitation of the current multimedia instruction sets. Lookup tables are a central element for low-level image and signal processing that is part of the hardware of any frame grabber but can be used for incoming data only [14]. With a lookup table any function can be implemented. Especially useful are lookup tables for dyadic point operations with two operands. With such a lookup table any dyadic operation—including multiplication, division, magnitude of a 2-D vector, etc.—can be computed.

3.5.2 Global transforms

In contrast to point operations, *global transforms* such as the discrete *Fourier transform* (DFT) (Volume 2, Section 3.3) compute each element of the transform from *all* elements of the input data. Nevertheless it is possible to implement such transforms efficiently by SIMD algorithms.

We will show this with the example of the 2-D fast Fourier transform algorithm. The 2-D DFT can be parted into a 1-D row and a 1-D column transform (Volume 2, Section 3.4.2). The key point then is that multiple rows and columns can be transformed in parallel. In essence this leads to an inner loop with the multiplication of a complex scalar with a complex vector. This operation can efficiently be implemented with the multimedia SIMD instructions. The real and imaginary part of the constant are loaded into the multimedia registers and multiplied with the vector. The MMX `pmaddwd` instruction can be used to perform one complex multiplication with four real multiplications and two real additions in a single clock cycle.

It is, however, still awkward to implement an FFT algorithm in 16-bit integer arithmetic. Either significant roundoff errors are introduced or block-floating techniques are required. Thus the recent extension of multimedia instructions to 32-bit floating arithmetic by AMD's 3DNow! or Motorola's AltiVec are very useful for algorithms such as the FFT.

3.5.3 Convolution

Convolution or linear shift-invariant filtering is one of the most important neighborhood operations in signal processing (Volume 2, Section 5.3). There are several ways to execute convolution operations efficiently with an SIMD architecture. We demonstrate one here. A 1-D convolution can be written as

$$g'_n = \sum_{n'=-R}^{R} h_{n'} g_{n-n'} \quad \text{or} \quad \boldsymbol{g}' = \sum_{n'=-R}^{R} h_{n'} S_{n'} \boldsymbol{g} \tag{3.3}$$

where the shift operator $S_{n'}$ shifts the vector $\boldsymbol{g}$ by n' elements. Thus the inner loop consists of the following basic operation

$$\boldsymbol{g}' = \boldsymbol{g}' + h_{n'} S_{n'} \boldsymbol{g} \tag{3.4}$$

A vector is multiplied with a constant and the result accumulated in another vector. This operation is repeated for all nonzero coefficients of the convolution sum. This way to execute a convolution operation is efficient as long as the vectors fit into the primary cache.

The true problem for this operation is caused by the shift of the input vector. The multimedia instructions require that the data are aligned on 64-bit boundaries. While it is easy to align the beginning of vectors and image rows at this boundary, the vector becomes dealigned because of the pixelwise shift. Thus additional shift operations are necessary with the pixels packed into the multimedia registers slowing down the overall performance of convolution operations.

3.5.4 Gray-scale morphology

Gray-scale morphology requires the computation of the minimum or maximum of vector elements for erosion and dilation operations (Volume 2, Chapter 21). With standard instruction sets, the computation of minima and maxima requires comparison operations and conditional branches.[1] As these branches cannot be predicted they permanently cause slow-down of the executing stream.

Multimedia instruction sets either use comparison instructions that generate masks for the computation of minima and maxima with logical operations or include these operations directly. The following example shows the MMX assembly code in the inner loop of a vector maximum routine.

Example 3.4: Maximum computation with MMX instructions

This routine uses the greater than comparison instruction `pcmpgtw` to mask the elements that are greater. This mask is used to cut out (`and` operation) the greater values in the first operand and the negated mask to cut out the greater values in the second operand. A subsequent `or` operation combines all maximal values. In the following code fragment of the inner loop of the maximum routine, eight 16-bit integer pixels from two vectors are processed per loop scan.

```
m1:
        movq    mm0, [esi]
        movq    mm1, [esi+8]
        movq    mm2, [edi]
        movq    mm3, [edi+8]
```

[1]Recently conditional move instructions have been added to standard instruction sets (e. g., for the PentiumPro and Pentium II processors) that avoid conditional branches.

```
movq    mm4, mm0
pcmpgtw mm4, mm2  // mm4 = 1st mask
movq    mm5, mm1
pcmpgtw mm5, mm3  // mm5 = 2nd mask
pand    mm0, mm4
pandn   mm4, mm2
por     mm0, mm4  // mm0 = maximum
pand    mm1, mm5
pandn   mm5, mm3
por     mm1, mm5  // mm1 = maximum
movq    [edi], mm0
movq    [edi+8], mm1
add     esi, 16
add     edi, 16
dec     ecx
jg      m1
```

3.5.5 Global segmentation

A *global segmentation* operation is also a point operation and as such can easily be implemented by the comparison instructions discussed in Sections 3.4.6 and 3.5.4. With the aid of shift operations it is also possible to generate a binary image in an efficient way.

It is not possible to accelerate edge-oriented segmentation operations with SIMD algorithms because such an operation is inherently serial. Compact codes for binary objects, such as the runtime length or chain code, likewise cannot be generated from image data with SIMD algorithms.

3.5.6 Binary morphology

Morphological operations on binary images require bitwise *logical operations* (Volume 2, Chapter 21). If binary images are stored with one bit per pixel, 32 pixels can be packed into a standard register and 64 or 128 pixels in a multimedia register. Thus operations with binary images are considerably faster than with gray-scale images because many pixels can be processed in one register in parallel. The acceleration factor is, of course, not equal to the number of bits in the registers, because misalignments of the bits require additional shift operations and the composition of a bit string from two bit strings. For standard 32-bit registers, it is still to be expected that binary images can be processed about 10 times faster than gray-scale images.

For the implementation of binary morphological operations with multimedia instruction sets an additional acceleration by a factor of two or four, depending on the register width of the multimedia registers as compared with the width of the standard registers, can be achieved.

3.5.7 Classification; neural networks

Finally, we discuss various *classification* algorithms and operations with neural networks. For the reasons discussed in Section 3.5.1, lookup table operations cannot be accelerated with SIMD instructions. Two other types of classification techniques are more suitable, *box classification* and *minimum distance classification*.

Box classification requires two comparison operations for each dimension of the boxes that model the clusters in feature space [14]. These comparisons can be performed in parallel and can thus be computed efficiently on packed data. Minimum distance classification is based on the computation of the distance between the P-dimensional feature vector $\boldsymbol{m}$ and the cluster centers $\boldsymbol{m}_q$:

$$d_q^2 = \left|\boldsymbol{m} - \boldsymbol{m}_q\right|^2 = \sum_{p=1}^{P} (m_p - m_q)^2 \tag{3.5}$$

This operation can also be implemented efficiently with SIMD operations provided that the dimension of the feature space is large enough.

The basic and computationally most expensive operation of *neural networks* is the accumulation of the weighted input values (Volume 2, Chapter 23):

$$g' = \sum_{p=1}^{P} w_p g_p \tag{3.6}$$

Mathematically it is equivalent to an *inner* or *scalar product* between the weight vector $\boldsymbol{w}$ of the neuron and the input vector $\boldsymbol{g}$. This operation can be computed efficiently in parallel using, for example, the multiplication-addition instruction (Section 3.4.3) of the MMX instruction set.

Usually, the output of the neuron is transformed by a nonlinear function and then used as a final output or the input for further neurons (Volume 2, Chapter 23). This is an operation that is typically performed with a lookup table and thus cannot be accelerated, as discussed in Section 3.5.1, by SIMD instructions.

3.6 Conclusions and outlook

This chapter concludes after an analysis of the possible peak performance of multimedia instructions in Section 3.4 and of the implementation of typical low-level signal processing algorithms on SIMD multimedia architectures in Section 3.5 with some results from benchmarks computed with the image processing software heurisko on various PC platforms. The benchmarks are also available on the CD.

***Table 3.6:** Performance of image processing operations optimized with MMX instructions. The performance is given in MPixels/s for both MMX disabled (-MMX) and MMX enabled (+MMX). All image sizes are 512×512*

Operation	Pentium 166 MHz		Pentium II 266 MHz	
	-MMX	+MMX	-MMX	+MMX
Addition/Multiplication C = f(A,B):				
Addition, 8-bit	11.0	49.3	77.5	143
Addition, 16-bit	7.4	23.4	24.3	32.4
Multiplication, 8-bit	6.0	38.8	65.4	121
Multiplication, 16-bit	5.5	23.3	23.7	32.5
Accumulation A = A+B:				
8-bit to 16-bit	9.8	29.9	49.5	78.7
8-bit to 32-bit	9.6	16.7	16.7	26.3
16-bit to 32-bit	8.4	14.0	16.2	22.3
Convolution:				
with 1/4[1 2 1]	2.4	22.7	8.4	42.6
with 1/16[1 2 1, 2 4 2, 1 2 1]	1.1	10.4	2.8	18.9
Laplace	1.3	10.4	5.0	28.6
Sobel	1.4	10.4	4.4	26.7
with 1/16[1 4 6 4 1]	1.4	15.8	4.6	29.4
with 5×5 mask	0.3	4.1	1.0	7.8
Erosion and Dilation:				
with 3×3 mask	1.5	10.7	4.1	23.6
with 5×5 mask	0.6	4.3	1.9	9.3

From Table 3.6 can be seen that up to 400 MOPS can be reached with modern standard processors. Thus, more complex real-time image processing (10 MPixels/s) with up to 40 operations per pixel—for example a convolution with a 5×5 mask—is possible.

In this chapter, modern multimedia architectures were discussed. The analysis showed clearly that low-level signal processing algorithms are accelerated well beyond the usual steady increase in computing power. This development was not initiated by the signal processing or computer vision community but by a new mainstream multimedia trend. However, as sophisticated algorithms are also required for sound processing and video compression, the architectures proved to be very suitable for all kinds of low-level signal and image processing algorithms. The continuous development of multimedia architectures indicates that further significant progress can be expected in this field, which will accelerate the computing power for signal processing faster than for general purpose number crunching. It can thus also be expected that some weaknesses of the current architectures (such as missing support of lookup table operations) will be removed and that more efficient handling of alignment problems and non-SIMD instruction types will be incorporated into future architectures.

3.7 References

[1] Borland International Inc., (1991). Turbo Assembler Quick Reference Guide.

[2] Intel, (1996). Intel Architecture MMX^{TM} Technology, Developer's Manual. Intel order No. 243010.

[3] Texas Instruments, (1988). Graphics operations. In *TMS34010 User's Guide*. Houston: Texas Instruments.

[4] Motorola Inc., (1991). Graphics unit implementation. In *MC88110 Second Generation RISC Microprocessor User's Manual*.

[5] Mou, A. Z. J., Rice, D. S., and Ding, W., (1996). VIS-based native video processing on UltraSPARC. In *Proc. Int. Conf. on Image Proc. ICIP'96*, Vol. II, pp. 153–156.

[6] Sun Microelectronics, (1995). Visual instruction set user's guide.

[7] Intel, (1996). Intel Architecture MMX^{TM} Technology, Programmer's Reference Manual. Intel Order No. 243007-001.

[8] Intel, (1996). Intel MMX^{TM} Technology Overview. Intel order No. 243081-001.

[9] AMD, (1996). AMD K-6 Processor Multimedia Extensions (MMX). AMD publication No. 20726.

[10] AMD, (1998). 3DNow! Technology Manual. AMD Publication No. 21928.

[11] AMD, (1998). AMD-K6-2 Processor Code Optimization. AMD Publication No. 21924.

[12] Motorola Inc., (1998). AltiVec technology programming environments manual.

[13] Rubin, P., Rose, B., and McCallig, M., (1996). Motion video instruction extensions for alpha. Digital equipment.

[14] Jähne, B., (1997). *Digital Image Processing—Concepts, Algorithms, and Scientific Applications,* 4th edition. New York: Springer.

4 Customizable Medical Image Processing Systems

Athanasios M. Demiris, Carlos E. Cárdenas S., and
Hans Peter Meinzer

Abteilung Medizinische und Biologische Informatik
Deutsches Krebsforschungszentrum Heidelberg (DKFZ), Germany

4.1 Introduction

4.1.1 Developing specialized components for a host system

The image processing functionality needed in a specific domain, for example, radiology, consists of an invariant and a variable part. All image processing functionality that aims at interpretation or manipulation of the semantics of the image make up the variant part of the system. This part changes for every new application in the domain. It is obvious that a radiology department will need a different set of image processing

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

tools for the analysis of *magnetic resonance*[1] (MR)-mammograms than for the volumetric calculations in *computer tomography* (CT) images of the upper abdominal area. In these examples the end users are clinical radiologists. Their requirements are expressed in terms of the clinical cases they want to treat and that is exactly the way in which the software solution offered to them should be addressed and organized by the developers. This obvious observation calls out for specialized modules of a system that target each different problem in an application domain even if there are functional overlaps among them. The fact that a region-growing algorithm probably would be used in parts of more than one case does not justify collecting all possible functions, integrating them into one *monolithic system*, and installing it in the site of an end user. This attempt to *a priori* cover all possible scenarios by structuring the end-user system in terms of image processing functionality increases the work load and decreases the acceptability. Such solutions cannot be integrated into everyday clinical routine. Creating specialized components addressing specific problems instead of all-purpose systems enforces user-centered design and developments as well as encapsulation of overcomplicated functionality, which is feasible only when a concrete and restricted application domain is addressed.

The invariant part of domain-oriented image processing systems deals with the image handling. Image handling means image retrieval, display, contents-preserving functionality like gray-level manipulations, zooming and similar visual information enhancements, storage and transfer. Although the core of each application should vary for each case this part should be uniform and used in an intuitive way. A detailed analysis of the work situation should be carried out and the resulting system should contain a domain specific visual vocabulary, that is, domain specific elements, terms, interactions etc. as shown by Gulliksen and Sandblad in [1]. If this *visual vocabulary* is formalized appropriately as shown in Section 4.4.4, other systems may adapt to it—thus resulting in a uniform look-and-feel for all applications of image processing in a specific domain or at least in one end-user site.

The invariant part of an image processing application may be used as a framework to host components addressing specialized issues. Thus a viewing station in a radiological department may be extended to include modules for the analysis of CT image-series of the liver, or digital mammography etc. Then the main task of the image processing specialist would be to extend an existing host system by a new module in a system-conforming way to address a new application. This gives software development for image processing applications a new dimension, which can be accomplished with the existing software technology. However, an infrastructure for this kind of development is needed.

[1]For magnetic resonance imaging see Volume 1, Chapter 22.

There are two central aspects in this domain-specific kind of software development that are not supported by conventional software engineering environments. First of all there are no systems that can adapt their outcome to conform with one or more given host systems. This is a task that has to be taken care of by a developer, thus leading to delays because of the time needed to become acquainted with the host system and the target characteristics, as well as the permanent limitations in the development of a solution. The second aspect is that the software components created or used for the assembly of the new application module can only be viewed, in such an environment, as yet another piece of software with no specific characteristics. The description of an image processing component as such contains many more characteristics than a generalized software component, which contributes numerous advantages, among them the easier retrieval of the correct component to use.

The latter is a more general problem of software development. *Code reuse* is supported by current programming paradigms, for example, object-orientation, but the problem that remains to be treated is the acquisition of the appropriate code to be reused. The same problem arises on higher abstraction levels as well, especially when trying to locate the appropriate set of software parts to reuse. This is a problem that can only be efficiently tackled in domain specific environments.

Since the generalized software development tools fail to successfully support the development of image processing components there is a need for dedicated solutions. The creation of new development environments for each branch is for a number of reasons not the optimal solution. The most important reason is that all developers have selected their preferred tools and put them together into a development environment. Introducing a new solution that can handle at least the same problems in a more domain-oriented way will definitely cause a steep learning curve and require many releases and person-years before it can be used in an efficient way. The person-years required will certainly include interdisciplinary knowledge ranging from compiler theory to advanced image processing and further to radiology or any other application domain.

We will introduce an alternative approach for the development of components for image processing systems by presenting an information architecture that can be used to extend existing development environments with domain specific parts. This *information architecture* (or recently more often referred to as *infotecture*) will itself be implemented as a series of components that can be added to any existing development system that follows the object-oriented paradigm.

The overall scenario for the application of object-oriented software engineering in computer vision applications is shown in Chapter 5. We use the concepts presented there to create components for imaging applications especially in the medical imaging domain. Components in our context can be thought of as software building blocks containing an algorithmic body along with a *graphical user interface* description and suggested parameter values for operation in a specific data context, for example, medical data. The algorithmic body can be created in any programming language. In Chapter 6, a generic solution for the creation of algorithms by means of parameterized types is presented. The use of such techniques enhance reuse not only at the component level, but also at the level of algorithmics in a specific programming environment (higher granularity reuse).

4.1.2 Requirements

An architecture for the support of image processing development should comply with a series of requirements. The need for a domain-oriented character of such an architecture has been emphasized in the previous section and is definitely one of the obvious requirements. Another quite obvious prerequisite is the reduction of "noise" during development, speaking in terms of information theory, that is, all activities not directly associated to the problem under consideration. This leads to a reduced *cognitive load* for the developer and should also lead to productivity enhancement. Many developers are very productive with a development tool after a long period of working with it. The learning curve for such systems is very steep. This is a reason for rejection of such a system. For the architecture presented here we declared minimal learning effort as one of the basic claims.

The solution to be developed should be considered as added-value software engineering for image processing and should coexist with existing development environments instead of replacing them. Because we decided to use innovative software engineering techniques this aspect can only be adhered to in part. Development systems that are not based or do not support such techniques cannot be supported by such an architecture.

As the major improvements of any development tool can only be carried out after the first release we proclaim the extensibility of the architecture itself as a major goal. The list of requirements for the architecture that should support the development of customizable image processing systems can be summarized as follows:

- inclusion of domain-oriented aspects;
- reduction of the cognitive load for the developers;
- optimal learning curve;
- coexistence with other development utilities;
- innovative software engineering concepts; and
- extensibility, adaptability.

4.2 State of the art

There are many options already available that support the development of image processing at various levels. At the lowest level there are the dedicated function libraries. There are obvious disadvantages inherent in this approach. The use of such libraries requires a comprehensive documentation that is often not given. Function libraries are a simple collection of algorithms with no meta knowledge apart from a written documentation, and thus not accessible to retrieving the appropriate algorithms. Capturing a sequence of algorithms in a function library covers only small parts of the development, leaving out interaction schemas, structural descriptions, etc.

At a higher abstraction level there are the *all-round development environments*. The amount of such tools has reached dimensions that make their archiving necessary (as seen in many professional sites in the World Wide Web). Valaer and Babb classified systems for the creation of user interfaces in their article [2] but their classification can be used very well for software development environments in general. These environments do not support domain specific development. Although the systems falling into the category of *Computer Aided Software Engineering* (CASE) environments promise a drastic reduction of development efforts they are not used widely due to many reasons ranging from complexity to compatibility as shown in Iivari [3].

There have been some efforts directed at creating round *domain-oriented development environments*. In the framework of the Advanced Informatics in Medicine (AIM) Program of the European Commission a group of European scientists designed an environment for the easy creation of ward information systems [4] containing an image processing related part [5]. The development environment covered all aspects needed for the creation of such a system, varying from natural language processing to advanced image processing and visualization. Innovative concepts were used and the overall scenario of distributed development was a forerunner of many later developments. The main disadvantage of that attempt was the complexity of the architecture. The output of the project was functionally successful but “oversized.”

A widely used category of tools for image processing applications are the so-called *data visualization environments* like AVS, Khoros and IDL. These systems are definitely helpful tools for the advanced user who is familiar with image processing and computer vision. They help to visualize data in a way that may be better evaluated but cannot lead to user-friendly software applications installed in end-user sites.

4.3 Identifying the development phases of a dedicated image processing module

In order to identify the components needed to add to existing development environments we start with the presentation of a possible workflow to be supported. This workflow has the goal of distinguishing all activities of a developer that are not yet supported by common environments and can be supported by a completing architecture.

Step 1. Inspect the new data. We assume that the starting point of the development process is the establishment of a contact between the end-user and the developer. The end-user expresses the wish for a new application for a specific domain. After the identification of the goals of the system the developer launches the production process. The first activity involves becoming familiar with the new data material. Therefore the developer might want to look up either anecdotal experiences of colleagues or information found in the literature. Subsequently a statistical evaluation (unary statistics mainly) could give a first impression of how the data compare to previous cases. The features statistically evaluated may vary from simple gray-level information to texture properties. Often, if the goal of the system is the segmentation of specific image parts, the developer manually creates a resulting image and tries to identify characteristics of the target object to be located compared to the surrounding areas.

Step 2. Select the appropriate algorithmic material. After the inspection of the image material the developer looks for possible algorithms in his/her "arsenal" or the ones of the team members that seem most appropriate for the task to be addressed. This requires good knowledge of image processing and documentation, preferably in a uniform way, of practical experience using algorithms from every developer. The selection of the algorithms can sometimes take place according to technical criteria concerning the implementation of the algorithm, especially when the developer has a clear idea of the way the problem has to be approached, and sometimes according to more abstract criteria, for example, all algorithms that detect areas etc.

Step 3. Apply the selected algorithms on the data. Algorithms that stand as candidates for integration in the final system need to be tested with the image material under consideration. The most important aspect is the selection of the optimal parameter values. Many image processing algorithms are highly parametric, which makes their application very difficult and the test phase relatively complicated. The tests are simplified when using a simple graphical user interface for experimenting with parameter values, as dedicated graphical user elements

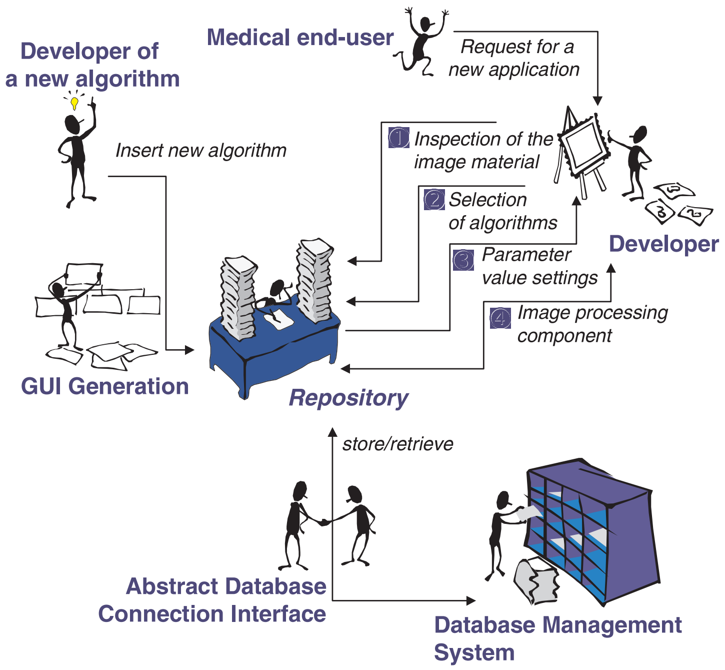

Developer of a new algorithm
Medical end-user
Request for a new application
Insert new algorithm
1 Inspection of the image material
2 Selection of algorithms
3 Parameter value settings
4 Image processing component
Developer
GUI Generation
Repository
store/retrieve
Abstract Database Connection Interface
Database Management System

Figure 4.1: *The development scenario to be supported by the component architecture.*

(e.g., sliders etc.) allow faster access, and the display of the resulting images is the only appropriate feedback for the effects of the algorithm application.

Step 4. Plugging algorithms together to create a module. When a satisfactory solution is achieved the resulting module needs to be integrated in a host application. Without any tool support this results in systems engineering in which the developer tries to adjust the results of the previous steps to a given system design. This last stage in the development cycle requires advanced knowledge of software engineering concepts, graphical user interface implementation, and ergonomics.

The foregoing steps were identified after a work situation analysis of image processing developers in our department and may vary from site to site. We describe them at a high abstraction level in order to cover a maximum of possibilities. These steps result in aspects that need to be addressed in a domain-oriented development environment. We identified on the basis of these steps the following components that need to be added to software engineering systems in order to sufficiently support the elaboration of a dedicated image processing module aimed at integration with a customizable host system.

4.4 Components of the architecture

The architecture can be subdivided into four basic components aimed at supporting the activities identified in the previous section. The "heart" of the architecture has to be a medium for the storage and retrieval of *image processing algorithms* and subcomponents in a flexible way. As this component contains a description of the image processing algorithms themselves (meta-information from the developer's point of view) used for their structured administration we call this component a repository.

The simple statistical evaluation of the image properties in order to become acquainted with the new data is a process we decided to support in the architecture. The reason is that most developers had to create new evaluation algorithms that would produce data-output usable by existing statistics systems. Such a component could offer a series of predefined statistical evaluation functions that could be activated or deactivated according to the needs of the developers.

The estimation of optimal parameter sets is a time consuming activity. The outcome of this activity is a valuable experience regarding both the used algorithmic and image material. We decided to support this activity by providing the developer with a facility for the storage and retrieval of parameter sets for certain algorithms and images and, furthermore, with the means to have initial values calculated automatically (when there are no stored values). The process of creating components for a customizable host system cannot be completed until a graphical user interface for the variation of the parameter values and the interaction with the host system is created. For this task we introduced a new component in the architecture dealing with the creation of such user interfaces based solely on the image processing algorithms with no extra effort required from the developer.

4.4.1 Repository

The central part of the proposed architecture is the *algorithm repository*. It is the means for storing and retrieving data in a structured way according to more than just technical criteria. The repository consists of two parts. The first part is the model describing the elements that may be stored in it. This description can be thought of as meta-information because it is information about the actual image processing algorithms. The second part maps the structure to a *database management system* using an *abstract database connection interface*.

There have been many attempts recently to create *meta-information facilities* and description formalisms, especially with respect to the description of object-oriented systems. The widely spread Microsoft repository enables developers to reuse software components in various development systems and environments, is based on Active-X technology, and uses SQL database mechanisms for the storage and retrieval of the components [6]. Unisys has developed the universal repository [7], which is compliant with the OMG-MOF standard.

There are also three standardization attempts in this area. The best known standard in the field of repositories and meta-information modeling is CDIF (Computer-Aided Software Engineering Data Interchange Format) [8]. CDIF evolved out of the need for the use of software pieces or systems created with different CASE Tools. Each CASE tool used to store the information about the software in a different format caused incompatibilities when porting from one CASE tool to the other. Thus the standardization committee tried to create a description for the handling of meta-information. Another standard aligned to CDIF is OMG-MOF (Object Management Group, Meta Object Facility) as described in [9]. The MOF is a CORBA-based facility concentrating on the exchange of meta-information. And finally there is the MDIS standard (Meta Data Interchange Specification).

All approaches and standardization attempts concentrate on a high level of abstraction; the goal is to capture the semantics of general purpose software development. We aim at the handling of specialized software in the image processing domain, thus a different approach to meta-information representation is needed. Apart from this fact, the approaches mentioned herein contain an inherently high degree of complexity.

We created a new model for image processing algorithms by specializing the concept of a *software building block*. We treat a software building block as a unit consisting of an algorithmic part and an accessing interface. A sequence of image processing algorithms can be viewed as a specialization of software building blocks. Each sequence of algorithms contains at least one image processing algorithm. Each algorithm consists of a signature and a body. The signature of an algorithm is the set containing the name of the algorithm, the parameter set, and the return value. By this information any algorithm can be identified in a unique way. That is the reason for the introduction of the term signature to describe this specific context. Every parameter, as well as the resulting value, may underlie given constraints regarding their value range.

Each algorithm can be applied in a specific context. A special form of an algorithmic context is the execution context, for example, the hardware platform for which an algorithm was made, the programming language the algorithm was implemented in, whether it is only a function, or a ready-to-use executable or an applet etc. Another specialized form of a context is the data context. Some algorithms can operate only on specific image data. An example coming from the medical domain would be the application of an algorithm on ultrasound image material.

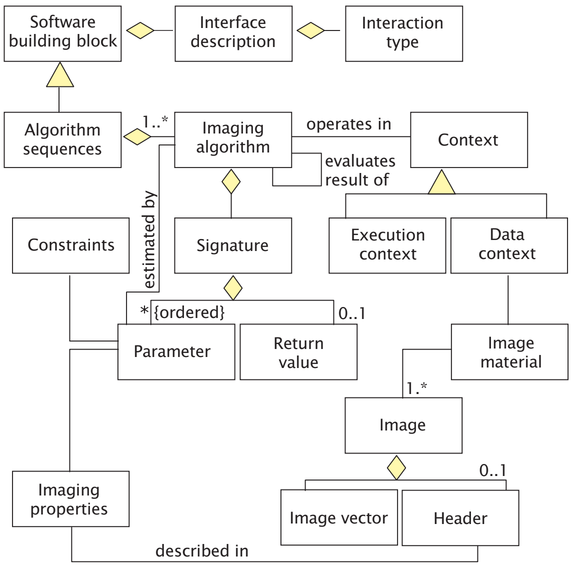


Figure 4.2: *A small part of the class diagram of the repository in the UML notation.*

Algorithms of that kind sometimes cannot be used on CT or MR images. There are algorithms that can only be applied on images if some characteristics are given, for example, contrast-enhanced imaging during acquisition.

The herein presented part of the model is shown in Fig. 4.2 in the UML notation [10]. The structure has been simplified in order to capture the most important aspects of the image processing description without exploiting too many disturbing details.

This description allows an easy query of algorithms according to many criteria. All properties of the classes shown in the model can be used as a keyword for a search. Thus they help the less experienced developers trace more functions in a shorter period of time and advanced developers “remember” relevant algorithms that do not belong to their “everyday toolkit.” The querying of the stored algorithms may be carried out in several ways. One way is to use an application programmer’s interface (API) and programmatically access and query the repository. An alternative approach is the use of a dedicated and well-designed query module that connects to the repository and enables the developer to transparently access the material stored and select the most appropriate parts.

There are three basic tasks that a developer wants to carry out when connecting to the repository. Apart from the query of the components there is also the need to introduce new components, update, replace, or remove existing ones. Therefore, the query module consists of three different task groups.

A vital aspect is the testing of the algorithms. Even after a successful search for algorithms according to the desired criteria, there should always be a way to access and test the components using visual feedback. Additionally, every time a developer enters a new algorithm into the repository a graphical representation for that algorithm should be created in order to call the newly entered algorithm a software building block according to our definition. We explicitly support the creation of a graphical user interface based solely on the declaration of the underlying algorithm and discuss the solution to this problem in Section 4.4.4.

The information structured by the model shown here needs to be persistently stored, that is, beyond the execution time of the system. Therefore, either the file-system or a database should be used. This would mean a mapping of the forementioned described model to a dedicated database model. We decided to avoid any proprietary solution by binding to a specific database provider by using an abstract database connection interface. Those chosen were the widely spread Object Database Connectivity (ODBC) by Microsoft [11] and the more recent JDBC for Java [12], which is a superset of ODBC, containing a so-called "bridge" to it. By using one of these possibilities (in our case the latter), the code to access a database management system remains unchanged even if the underlying database is replaced by another one. The one prerequisite for such flexibility is the support of the CLI standard (Call Level Interface introduced by X/Open) on the side of the database providers. However, most providers support both possibilities. The basic reason for the use of an abstract database connection is the existence of databases in image processing groups, which could be used for the repository as well.

4.4.2 Statistical image analysis component

When new image material needs to be analyzed or a different problem needs to be solved, the data to be used has to be examined in order to identify collective trends that might lead to a generalized solution. This approach is very common, especially when the data has to be segmented. The developers try to identify unique characteristics of the object to be located or all criteria that will lead to a separation from the surrounding pixels or voxels. This is a procedure that can be supported by a development environment.

In order to help the developer gain a first impression of the consistency of the data, the proposed architecture is expanded by a component that offers customizable data exploration by means of unary statistics. The task of this component is rather trivial and concentrates on the simple and easy application of selected measurements and an adequate presentation of the results.

Several criteria of the image scenery can be evaluated. The component can be configured to calculate a series of characteristics (e.g., the mean or median gray value, the bounding box or the center of inertia of binary objects, or the numeric result of virtually every evaluation algorithm that is contained in the repository etc.) and produce an output that is adequate for displaying or further processing with the most common statistical systems or spreadsheets. The component can apply the measurements on native or preprocessed data. All operations on the data are done to avoid erroneous assumptions. An example of use is the investigation of contrast-enhanced CT images of the liver. In order to identify invariant characteristics of the liver a small collective of image series was preprocessed, that is, manually segmented into liver and non-liver areas. The statistical evaluation component was configured to measure the variation of the bounding box, the actual volume of the segmented liver tissue etc. in order to obtain measurements for subsequent processing.

4.4.3 Optimal parameter value estimation component

One of the most time-consuming activities of an image processing developer after the selection of possibly applicable algorithms is the estimation of the optimal parameter values for the specific image material under consideration. This task requires extensive testing and experimentation with different parameter settings.

The work presented here is very much dependent upon a given domain, for example, the medical imaging domain. Other parts of the architecture presented here can be easily transferred to virtually any computer vision application, either as is or after slight modifications. The component for the parameter-value estimation requires domain-specific knowledge that makes it almost impossible to keep it when migrating to other application areas as presented in this volume. For that purpose the complete parameter-value estimation component needs to be replaced.

One possibility is to lookup existing values in case the algorithm had already been applied to similar image material. We introduced a link between the algorithm and the image material in the repository model in order to be able to store information on the known uses and restrictions of the stored algorithm. This is a way of capturing valuable image processing experience in a formalized way. The retrieval of the appropriate parameter sets, if available, takes place in a domain specific way, that is, the criteria for the lookup are obtained from the descriptive fields of the image header, which is assumed to be in the DICOM format (Digital Imaging and Communications in Medicine as described in [13]). Although many vendors of medical imaging equipment still do not fully support the DICOM standard introduced by the National Electrical Manufacturers Association (NEMA) the tendency is towards a broader use of the standard in more and more medical imaging devices. A standard conform header contains information regarding the imaging modality, the scenery, the conditions of acquisition, etc.

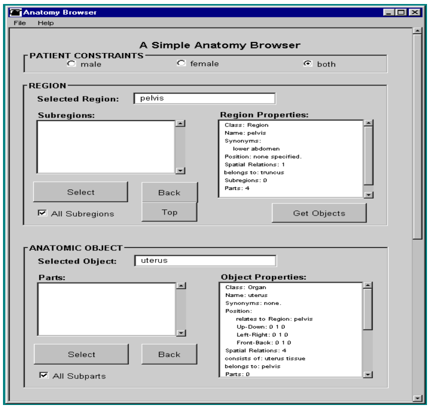
Unfortunately in the majority of the cases there is no information available about optimal parameter values for a specific algorithm when applied to given image material with a defined purpose (e.g., the optimal settings for a region-growing algorithm for the detection of an organ in T1-MR images using a defined imaging sequence). Therefore a possibility for the quick estimation of initial parameter values should be offered to the developer. This problem has been tackled by many research teams especially in the area of knowledge-based image understanding. Our goal is to roughly estimate a parameter set that will serve as a good starting point for a given task.

We developed a radiological information base that describes the properties of anatomical regions in medical images [14]. In Fig. 4.3a graphical query component for the retrieval of radiological and imaging knowledge stored in the database is shown. Therefore a model similar to the one in the repository was developed, and all known values and properties for organs in medical imaging modalities were stored in the database. When a developer enters a new algorithm in the repository he/she has the option to link the parameters of the algorithm to imaging properties found in this database. A typical example is a gray-level region growing algorithm. The selection of the upper and lower gray values affects the size of the object. If the developer tries to use an algorithm for the detection of an organ, for which the links between parameters and imaging properties have been established, the system can assist the developer by making a first suggestion. For this purpose an arbitrary set of values is calculated within the allowed range for each parameter and the algorithm is applied on the image. The resulting object is compared to the expected organ using a simple heuristic fuzzy evaluation schema that evaluates the following aspects:

- size of the segmented object relative to the borders of the body;
- center of inertia of the segmentation result;
- extent of the segmented object;
- orientation;
- connectedness (holes in the segmentation result); and
- homogeneity (gray-value based).

If the result does not satisfy the criteria formulated in terms of fuzzy rules, than a new parameterization is launched. The values are modified according to the results of the forementioned measurements. Thus

a



PATIENT CONSTRAINTS: male / female / [✓ both]

REGION

Selected Region: pelvis

Subregions:

Region Properties:

```
Class: Region
Name: pelvis
Synonyms:
   lower abdomen
Position: none specified.
Spatial Relations: 1
belongs to: truncus
Subregions: 0
Parts: 4
```

[✓] All Subregions

ANATOMIC OBJECT

Selected Object: uterus

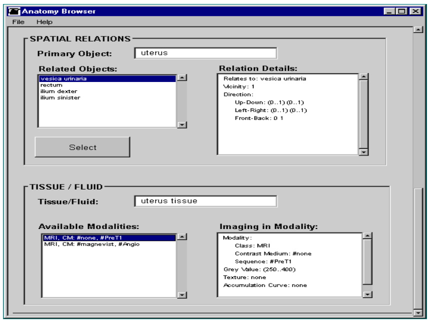
Parts:

Object Properties:

```
Class: Organ
Name: uterus
Synonyms: none.
Position:
    relates to Region: pelvis
    Up-Down: 0 1 0
    Left-Right: 0 1 0
    Front-Back: 0 1 0
Spatial Relations: 4
consists of: uterus tissue
belongs to: pelvis
Parts: 0
```

[✓] All Subparts

b



SPATIAL RELATIONS

Primary Object: uterus

Related Objects:

```
vesica urinaria
rectum
ilium dexter
ilium sinister
```

Relation Details:

```
Relates to: vesica urinaria
Vicinity: 1
Direction:
    Up-Down: (0..1) (0..1)
    Left-Right: (0..1) (0..1)
    Front-Back: 0 1
```

TISSUE / FLUID

Tissue/Fluid: uterus tissue

Available Modalities:

```
MRI, CM: #none, #PreT1
MRI, CM: #magnevist, #Angio
```

Imaging in Modality:

```
Modality:
    Class: MRI
    Contrast Medium: #none
    Sequence: #PreT1
Grey Value: (250..400)
Texture: none
Accumulation Curve: none
```

Figure 4.3: ***a*** *The graphical interface for the region-oriented search in the radiological database and* ***b*** *the spatial relations of a selected object as well as its imaging properties for a given modality, which can be queried by means of an API as well (see also `/movies/movie1.mov`).*

if the extent of the object and its relative size are not compliant with the expected values to a very high degree, the upper and lower gray limits will be modified accordingly. The procedure is sketched in Figure 4.4.

The latter way of estimating optimal parameter values is a heuristic approach that has not yet been proven to be effective in a sufficient number of cases. Its experimental character is accompanied by another disadvantage, that is, the lack of enough heuristic rules for all imaging modalities and anatomic regions.

4.4.4 Creation of graphical user interfaces

As mentioned, one of the central aspects of the selection of the appropriate algorithmic material is the ability to test visually the effects on the data under consideration. Therefore *graphical user interfaces* (GUI) are needed. Developers of medical image processing systems are often not familiar with the process of creating user interfaces, which is a very complicated and time-consuming one. Thus many times the developers concentrate on the creation of any interface to their functions, which is a complicated process, ignoring basic ergonomics.

This is a very important stage in the support process of the developer as this is the aspect for which a maximum gain is expected. The goal is the creation of a configuration interface for each component; by that we mean an interface for obtaining the values of the parameters from the image or an entry of the user and a facility for applying the algorithm. An automated support is feasible. There have been many

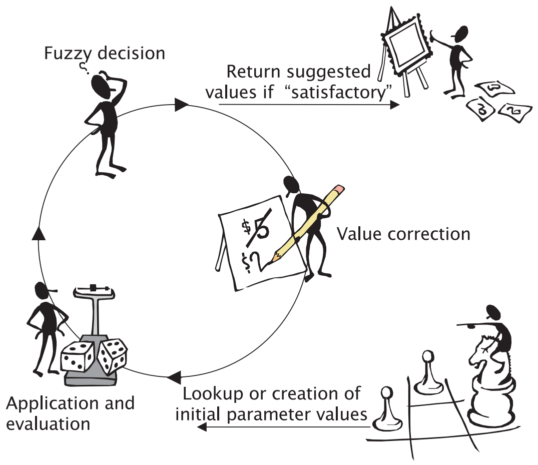

Fuzzy decision
Return suggested values if "satisfactory"
Value correction
Application and evaluation
Lookup or creation of initial parameter values

Figure 4.4: *After the initial values are applied, a series of measurements are carried out (evaluation) and combined by means of a simple fuzzy inference in order to decide on a recalculation of the parameter values or termination.*

attempts to automate the system generation and create a complete system on the basis of a semiformal or even a natural language description such as in [15] and [16]. The aim of existing research work is towards general purpose systems and concentrates on the automation of the software system generation as in [17].

An algorithm is applicable through its signature. Thus the signature can be used as the basis for an automatic creation of a user interface description for any imaging algorithm. We parse the source code file of the algorithm and detect the declaration line. Then the declaration is parted into a name, a return value, and the parameter list. The types of the parameters have to be known to the system and there should be a parser for the given language. For each type a description is created, for example, integer, real, binary, image, file-descriptor, text, array etc. This description can be converted using a lookup table to associate the description with a given element. Thus for the display and manipulation of numeric types, both integers and reals, one could use a text field displaying the current value for the parameter with two arrows for increasing and decreasing the value (the increment is different for a real and an integer). The final creation of the graphical elements can be done using a default set or according to a description for the target system. This means that any system that wants to retrieve image processing algorithms from the repository and integrate them as specialized modules should provide the repository with a description of its look-and-feel. This description is the code for the creation of these elements in a binary dynamic link format and can be implemented in any programming language.

The creation of dedicated elements for each parameter is a very straightforward approach and does not take into consideration important aspects of user interaction. If, for instance, a series of parameters describe the seed point of a region growing algorithm, then it would be very inefficient to ask the user for manual input instead of a mouse click in the image series that would obtain the values in a much more efficient way. Therefore we introduced the concept of an interaction pattern. We call an interaction pattern the way the user interacts with the algorithm configuration interface in order to pass the parameter values. We identified the three basic interaction patterns as follows: the simple interaction consists of value entries using dedicated elements. The interaction with multiple sources consists of some value entries using dedicated elements and some entries obtained by a mouse-click in the image. The third interaction pattern supports a combined interaction as the second pattern, extended by the possibility to use multiple input images.

For an algorithm like the Sobel operator the simple pattern would be sufficient. A region-growing algorithm requires a seed point in the image. For such an algorithm the option of using mouse interaction is very valuable. Logical combinations between images, such as the AND operation, require a series of input images in order to create an output. For this case the third interaction pattern should be used. In Fig. 4.5 these cases are displayed in a stand-alone version of a test editor. This test editor can be launched when the interface is created in order to test whether the automated creation procedure completed successfully or in order to test the application of the algorithm on the given data.

The interaction patterns as well as the types that the system can handle can be extended with not much overhead due to the way the system is implemented as discussed in Section 4.5.1.

The creation of the interface can be carried through either when the algorithm is inserted in the repository, or when it is being retrieved, if the interface description does not already exist. Should the algorithm be implemented in a programming language other than the ones known to the parsing module of the component or contain types that cannot be associated to graphical descriptions, it is always possible to use a manual configurator, which will create elements on demand out of a predefined palette.

The information stored in the repository is a meta description of what elements should be used for the creation of the interface. There is a conversion mechanism to the look-and-feel of any host application. Every host application implements a different look-and-feel, for example, use of different elements for the modification of numeric values, support of mouse-clicks in the image or not, coordinates returned after

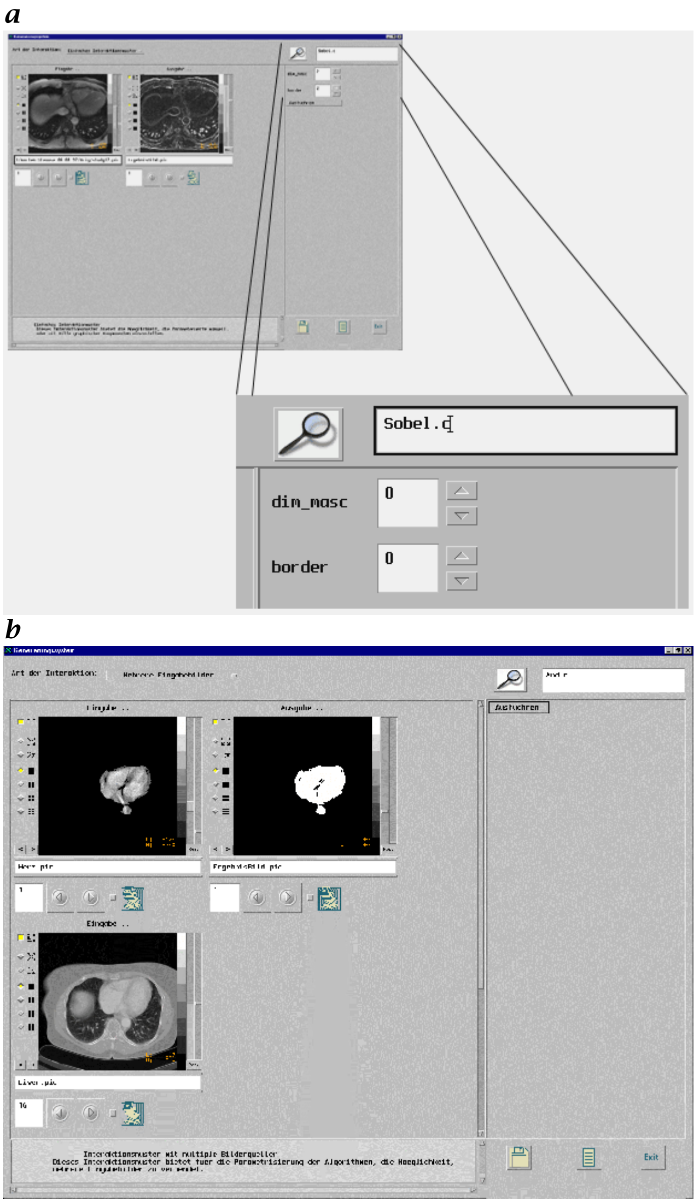


Figure 4.5: *A stand-alone test editor hosts the result of the GUI generation. In **a** the created interface for the Sobel operator is displayed using the simple interaction pattern while in **b** there are two images displayed for the AND operator.*

a mouse-click etc. In order for the repository to be able to convert to a target look-and-feel it is necessary to provide it with a description of the elements and the interactions supported by the host application. Otherwise the meta- description will be converted into the default look using the simple interaction pattern.

4.5 Implementation of the architecture

All components of the architecture are implemented in different stages ranging from an "alpha-release" of the parameter estimation component to the advanced version of the creation of the user interfaces. Isolated tests for each component of the architecture were carried out at our department.

For all components we used object-oriented technology and programming languages. The repository is currently implemented in Smalltalk and is accessible through a low-level TCP/IP interface (socket connection). For the time being we port it to Java in order to take advantage of Web-based accessing (querying) and the abstract database connection interface. All other components are implemented in C++.

4.5.1 Implementation techniques

One of the prerequisites for the architecture was extensibility and configurable use. The best possible implementation was in form of object-oriented frameworks. Frameworks are the advancement of the procedural libraries followed by the class libraries and the event-loop systems; these have been around for longer than a decade but became more popular in the past few years after the boom of object-orientation and resulted in a series of numerous frameworks as shown in [18]. With *object-oriented frameworks* a new way of program development was introduced by providing more than a collection of functions like the procedural libraries and a set of classes and they went further than the event loop-systems that supplied the developer with restricted predefined functionality.

Object-oriented frameworks are object collection with integrated interactions and cover the generic parts of given domains; in this context we use the term domain for both application domains, for example, desktop publishing, financial administration etc. and software development aspects that include parallel and concurrent programming, graphical user interfaces etc. The developer can use a framework (depending on its implementation) as is or configure it to match better the specific requirements of his/her tasks. Configuration can have different meanings starting from the subclassing and ranging to the replacement of entire parts. The ways the frameworks can be expanded are predefined. The parts of a framework that are used for its configuration are called hot-spots and their identification is crucial to the flexibility of the framework as defined by Pree [19].

Three components of the described architecture are implemented as frameworks in C++:

- The graphical user interface generation component can be extended to include more datatypes than are currently available. The parser is currently identifying algorithm-signatures in C files (and partly C++). Through subclassing further languages can be processed as well.
- The statistical evaluation component can host any number of evaluation techniques. The created output supports currently only a limited number of spreadsheet and statistical systems. The extension in the given hotspots can extend the possible generated outcomes of the evaluations.
- The parameter estimation component can handle specific rules defined by means of a simplistic fuzzy inference engine implemented in Smalltalk. The inference mechanism is also easily extendible to cover further formalisms.

For the creation of the frameworks we followed the development proposal made by Roberts and Johnson [20]. In their work they identify the tasks needed for the successful creation of object-oriented frameworks in terms of a so-called pattern language [21].

4.5.2 A sample use of the architecture

We used the interactive segmentation and volumetry system VolMeS, which was developed in our department, as an application scenario for the architecture. The reason for this choice was the complexity level of the system: the system uses sophisticated image handling mechanisms although presenting itself to the user as an easy-to-use system. The image processing algorithms in the system are treated as distinct instances and can be configured. The ergonomic criteria used for the development of the user interface lead to a well-defined user-interaction schema. The complexity is not too high for a first feasibility study though, because VolMeS is implemented as a single-process system restricted to the segmentation and volumetric measurements of medical image series and avoids visualization and communication aspects, which we wanted to ignore in the first release of the architecture implementation.

The system is parted in distinct functional areas as can be seen in Fig. 4.6. In the upper part a navigational and selection element common to many medical imaging systems is displayed. In the middle area there are three parts to be seen. The leftmost area displays DICOM information if contained in the image material. The central area is the area selected for processing an image out of the image-series. In the right part a set of tools is displayed. The toolbox consists of two semiautomatic and two manual algorithms: region growing using input from the image and dedicated entry fields, a correction algorithm based on the concept of illusory contours [22], a free-sketch "pencil," and an "eraser." Furthermore there is an option for setting a reference point in the image, a morphological operation for the instant filling of segmentation wholes, a volumetric component, and a possibility to launch custom scripts. Each of the tools operates using a set of default parameter values, which can be modified by selecting the "configure" option in a context sensitive menu displayed with every corresponding tool-button. The tool-area is replaced by a configuration area and the dedicated graphical elements for the input of new values are displayed. Both possibilities are shown in Fig. 4.6

Our defined goal as a first test for the architecture was to replace the algorithms of the toolbox after a query. Due to the lack of a graphical interface for the querying (see Section 4.7), the access of the repository and the querying has to be carried out using the dedicated API and the TCP/IP connection. The imaginary scenario is the adaptation of the tool to ultrasound images, which can hardly be processed by the given palette of tools. A query is launched with two search criteria:

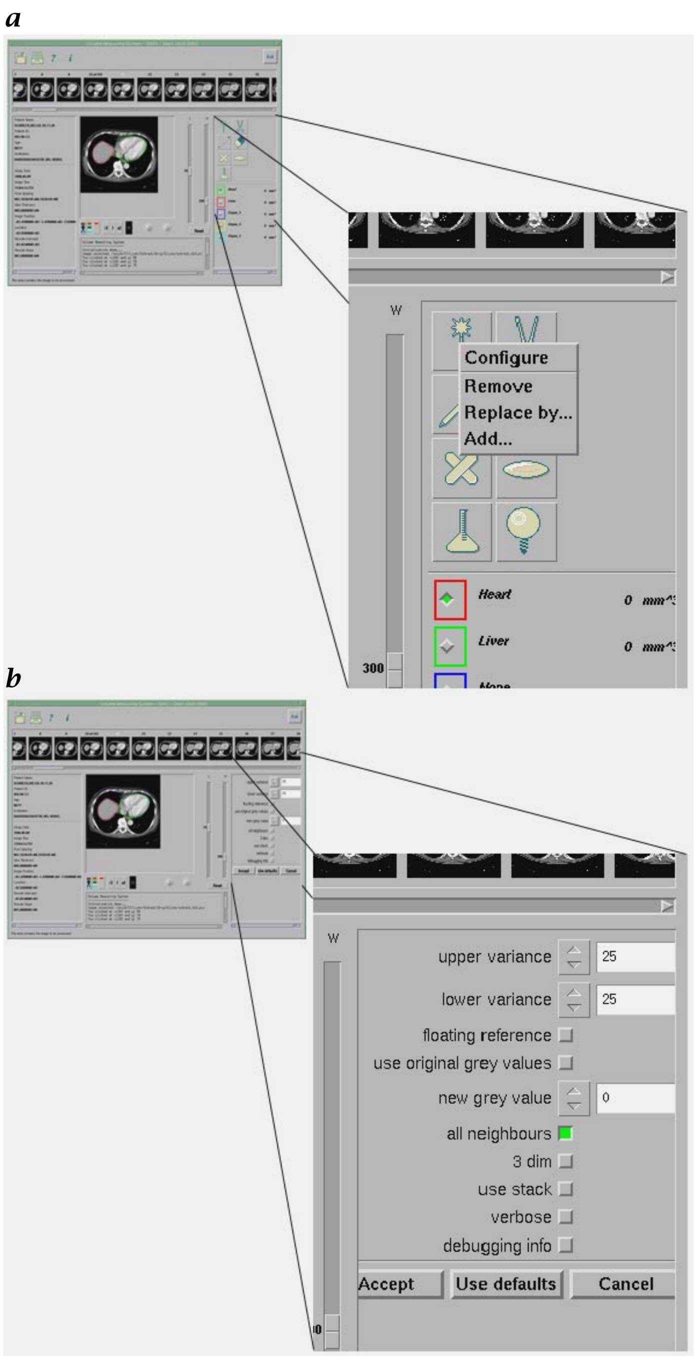
> find all segmentation algorithms for which it is known that they operate on ultrasound images.

The corresponding segment sent over the net to the repository has the structure:

> openQuery, locateAlgorithm, startCriteria, {, function, segmentation, }, {, imageCategory, US, }, stopCriteria, asHandle, closeQuery

It is obvious that the API for even such an easy query is rather complicated (see Section 4.7). After that a list of components is returned. One or more of the candidates can be selected for a given execution context. The selection will be passed to the system as a dynamic link library along with the graphical user interface in the correct look-and-feel. For the latter a description in a repository-specific meta language should exist for the system under consideration. If it does not exist the returned component can have its parameters modified only by the use of dedicated elements (not by mouse clicks or similar interactions) and displays in the repository-default look.

Furthermore an initial parameter estimation for the new algorithm could be carried out if the goal of its application was declared and known to the repository, for example, determination of the left ventricle in MR images of the heart, where a rough description of the left ventricle and information about the imaging modality should exist.



upper variance 25
lower variance 25
floating reference ☐
use original grey values ☐
new grey value 0
all neighbours 🟩
3 dim ☐
use stack ☐
verbose ☐
debugging info ☐

Figure 4.6: ***a*** *The simple customizable host system VolMeS in its work status and in the* ***b*** *parameter configuration status (see also* `/movies/movie3.mov`*).*

4.6 Conclusions

The work presented here is ongoing research dealing with the development support of specialized components for customizable image processing systems emphasizing medical applications and aiming at the demonstration of advantages of component-based development for customizable, domain-oriented image processing systems. The main goal was the development of parts that can be added to existing development environments in order to enhance their suitability for the image processing domain and better support the work of image processing specialists.

As already mentioned in Section 4.4.3, all components can easily be adapted to other application domains by simple extensions and introduction of the appropriate subclasses. The only exception is the parameter-value estimation component, which needs to be completely replaced due to its highly specialized character (the estimation of an optimal value for a parameter can only take place if the meaning of the parameter is modeled and that can only happen in a very domain-specific manner).

4.7 Future work

We are now concentrating on the integration of all modules and the statistical evaluation of the architecture. Our goal is the measurement of development speed-up after the introduction of the architecture. There are still many open issues in both the scientific and the technical part of the architecture. In this section we will present some future work with respect to each component.

Repository. The query component should itself be implemented along with a user interface as a dynamic linkable object in order to be callable from any host application with reduced programming overhead. For the time being we are developing a Web-based access to the repository. The repository implementation will migrate to Java. In order to replace the rather cryptic and nonstandardized query API (as partly shown in Section 4.5.2) along with the low-level accessing on the ground of TCP/IP sockets we are considering the use of CORBA [23] for distributed accessing of the repository data.

Statistical evaluation component. More output formats should be integrated into the framework.

Parameter estimation component. An extensive validation study is planned: two applications, the analysis of CT, MR and PET images of the liver, and EBT and MR images of the heart will be used in the parameter estimation for the vast majority of the algorithms stored in the repository for validation of the actual gain when using the component.

Graphical user interface generation. We have experimented with the development of an automated style-guide, and a more controlled development of *graphical user interfaces* by reducing the degrees of development freedom at run-time according to basic ergonomic rules as shown in [24]. This work could extend the architecture by another component, which this time would aim at the developer of the customizable host system and not the developer of specialized components.

Acknowledgments

The authors wish to thank Marc R. Goepfert, Antje Schroeder, Dr. Gerald Glombitza, and Dr. Manuela Makabe for their continuous support and contributions. We also wish to thank M. Merdes for his support during the creation of the manuscript.

4.8 References

[1] Gulliksen, J. and Sandblad, B., (1994). Domain specific design of user interfaces. *International Journal of Human-Computer Interaction*, **7**(1): 135–151.

[2] Valaer, L. and Babb, R., (1997). Choosing a user interface development tool. *IEEE Software*, **14**(4):29–39.

[3] Iivari, J., (1996). Why are CASE tools not used? *Commun. ACM*, **39**(10): 94–103.

[4] Degoulet, P., Jean, F., Meinzer, H., Engelmann, U., Baud, R., Rassinoux, A., Jagermann, C., Sandblad, B., Cordelle, D., and Wigertz, O., (1994). The HELIOS medical software engineering environment. *Computer Methods and Programs in Biomedicine*, **45**:91–96.

[5] Engelmann, U., Meinzer, H., Schröter, A., Günnel, U., Demiris, A., Schäfer, M., Evers, H., Jean, F., and Degoulet, P., (1995). The image related services of the HELIOS software engineering Environment. *Computer Methods and Programs in Biomedicine*, **46**:1–12.

[6] Bernstein, P., Sanders, P., Harry, B., Shutt, D., and Zander, J., (1997). The Microsoft repository. In *Proc. of the 23rd VLDB Conference, Athens, Greece*. San Mateo: Morgan Kaufmann.

[7] Unisys, C., (1996). *Universal Repository Information Model: Technical Overview*. Technical Report 86002094, Unisys Corp.

[8] CDIF, (1995). *CDIF—An Overview*. United Kingdom: The Stationery Office.

[9] Crawley, S., Davis, S., Indulska, J., McBride, S., and Raymond, K., (1997). Meta-information management. In *Proc. of the Formal Methods for Open Object-Based Distributed Systems (FMOODS) Conference, Canterbury, United Kingdom*. London: Chapman & Hall.

[10] Fowler, M. and Scott, K., (1997). *UML Distilled: Applying the Standard Object Modeling Language*. Reading, MA: Addison-Wesley.

[11] Geiger, K., (1995). *Inside ODBC*. Unterschleissheim, Germany: Microsoft Press.

[12] Reese, G., (1997). *Database Programming with JDBC and Java*. Cambridge, USA: O'Reilly.

[13] Hindel, R., (1994). *Implementation of the DICOM 3.0 Standard—a Pragmatic Handbook*. Chicago: RSNA.

[14] Link, J., (1996). *Entwicklung und prototypische Implementierung eines anatomisch-radiologischen Modells zur Unterstützung der medizinischen Bildanalyse*. Technical Report 82/1996, Div. Medical and Biological Informatics, Deutsches Krebsforschungszentrum, Heidelberg, Germany.

[15] Szekely, P., Luo, P., and Neches, R., (1993). Beyond interface builders: Model-based automated generation of user interfaces. In *Proceedings of Human Factors in Computing Systems, INTERCHI '93, Amsterdam*. New York: ACM Press SIGCHI.

[16] Puerta, A., (1997). A model-based interface development environment. *IEEE Software*, **14**(4):40–47.

[17] M.R. Frank, J. F., (1993). Model-based user interface design by example and by interview. In *Proceedings of UIST'93, ACM Symposium on User Interface Software and Technology, Atlanta, Georgia USA*. New York: ACM press.

[18] Lewis, T. (ed.), (1995). *Object-Oriented Application Frameworks*. Greenwich, USA: Manning Publications.

[19] Pree, W., (1997). *Komponenten-basierte Software-Entwicklung mit Frameworks*. Heidelberg, Germany: dpunkt.

[20] Roberts, D. and Johnson, R., (1996). Evolving frameworks: A pattern language for developing object-oriented frameworks. In *PLoPD3, Proc. of the PLoP '96*, R. Martin, D. Riehle, and F. Buschmann, eds.

[21] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., (1995). *Design Patterns: Elements of Reusable Software Architecture*. Reading, Massachusetts USA: Addison-Wesley.

[22] Glombitza, G., Makabe, M., and Meinzer, H., (1996). Formorientierte Korrektur von regionenbasierten Bildanalysemethoden. In *Workshop Digitale Bildverarbeitung in der Medizin, Freiburg, Germany*, B. Arnolds, H. Müller, D. Saupe, and T. Tolxdorff, eds., pp. 120–125. Berlin: Universitätsklinikum Benjamin Franklin, Freie Universität Berlin.

[23] Object Management Group Inc., (1995). *Common Object Request Broker Architecture and Specification, CORBA Revision 2*. Technical Report, Object Management Group Inc., Framingham, MA.

[24] Demiris, A. and Meinzer, H., (1997). Cognition-based development and evaluation of ergonomic user interfaces from medical image processing and archiving systems. *Medical Informatics*, **22**(4):349–358.

5 Software Engineering for Image Processing and Analysis

Dietrich Paulus, Joachim Hornegger, and
Heinrich Niemann

Lehrstuhl für Mustererkennung
Universität Erlangen-Nürnberg, Germany

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

5.1 Introduction

Configuring or programming image processing systems is a time-consuming task that requires specialized knowledge on the effects of image processing algorithms as well as knowledge about the implementation and interfaces. Clearly, *software engineering* is required for the application programmer of image processing systems. But even those who do not implement their applications themselves occasionally have to face software engineering problems. Several of the commercial or free packages for image processing that exist provide routines which can be plugged together to more complex operations. This does not solve software engineering problems; rather it shifts the basic building blocks to a higher level. The larger the application which uses such libraries, the higher is the importance of well-structured software.

The major problem in design of general imaging systems is that on the one hand highly run-time efficient code and low-level access to hardware is required, and that on the other hand a general and platform-independent implementation is also desired which provides all data types and functions for at least intermediate-level processing, such as results of segmentation.

Software reuse is crucial in any large system; well-documented packages should be usable even across applications. If every programmer is allowed to reprogram existing code, several pieces of code that serve the same purpose will soon be scattered around in the system.

Today's software engineering is closely coupled to ideas of object-orientation. Theoretically and practically, object-oriented programming gains a great deal of attention. *Object-oriented programming* can help simplifying code reuse; if applied properly, it unifies interfaces and simplifies documentation by the hierarchical structure of classes. A key mechanism of object-oriented programming is *polymorphism* [1]. In this chapter we give examples of how polymorphism can simplify image processing programs and how it can maintain the required efficiency.

We describe the use of object-oriented programming for the implementor of image processing and analysis software. In Section 4, a similar issue is shown with an emphasis on the configuration of an application system. In Section 6, an alternative to object-oriented programming is shown by the introduction of independent generic modules, which can also be used in combination with object-oriented programming.

We start with a short introduction of the general terminology of object-oriented programming in Section 5.2. Software developers for image processing today have the choice of several programming languages suited to their needs, for example, C, C++, Java, Ada, Fortran, etc. (Section 5.3). In general, image analysis has to use knowledge about the task domain. We outline an object-oriented implementation of a knowledge based image analysis system in Section 5.4; In this section we emphasize implementation issues, whereas in Volume 2, Chapter 27 the general structure is described formally. In Section 5.5 a novel architecture for classes representing data, actions, and algorithms for image processing is presented. This architecture is applied to segmentation, object recognition, and object localization. A summary follows in Section 5.6.

5.2 Object-oriented software engineering

Object-oriented programming has become popular in many fields including imaging applications. We briefly introduce the important ideas and terms of object-oriented software and the basic principles for object-oriented analysis, design, and programming. We discuss more specifically those software engineering issues that are relevant to image processing.

5.2.1 Object-oriented principles, analysis, and design

The object-oriented programming style suggests the decomposition of the problem domain into a *hierarchy of classes* and a set of communicating objects, which are instances of classes. The object-oriented programmer specifies *what* is done with the objects; the procedural way of programming uses aspects of *how* something gets done. One advantage of object-oriented software design is the one-to-one assignment between concepts in the application domain and the objects in the program. Even the analysis of the problem domain has to be involved in this mapping. Analysis and program design are no longer separated in the software development process; object-oriented analysis and design share the same terminology and tools. The first phase of any software development is to define the requirements. Three other connected stages, to be described in the following sections, are common to object-oriented software development. The most important ideas of object-oriented software that we introduce in the following sections are *objects*, *classes inheritance*, and *polymorphism*.

In the *object-oriented analysis* (OOA) stage, concepts of the *problem domain* and their correspondence are identified and specified. These *objects* are grouped to *classes*. Hierarchical relations between these classes are used; information that can be shared by several special classes will be included in a general class and passed to the special cases by *inheritance*. Objects are decomposed into their components, which are again described as classes.

Example 5.1: Object-oriented analysis

A typical problem domain from the area of image analysis is the recognition and localization of industrial objects on an assembly line. Concepts of this domain are the various parts, the belt, or nonphysical terms like speed, motion, or a stable position for the object. A hierarchical order of parts may group the parts according to their purpose. General terms used for a group of concepts can be identified.

In the *object-oriented design* (OOD) phase the attention shifts slightly towards the implementation domain. The conceptual class hierarchy created in OOA is overlaid with links that are meaningful for the implementation only. This causes a transition from the problem domain to the *solution domain*.

Ideally, two hierarchies are used. One relates the classes specific to the application domain, which were drafted in the analysis phase. The other hierarchy provides the implementation concepts, like sets, lists, or geometric objects. These two hierarchies are linked together, possibly creating multiple inheritance relations.

So-called *methods* are defined for the new classes. These methods provide access to the data represented in the classes and also perform the intended transformations on the objects.

Example 5.2: Object-oriented design

In Example 5.1, the classes for the industrial objects can now use geometric object classes to describe their shape, for example, a wheel will use a circle class. This combines the specific application with general definitions that are independent of the application.

Several graphical representations and mechanisms have already been proposed for OOA and OOD. Booch, Jacobson, and Rumbaugh combined their efforts and created the "*Unified Modeling Language*" (UML), which includes three essential parts [2]:

- guidelines for the vocabulary,
- fundamental modeling concepts and their semantics,
- notation for the visual rendering of the concepts.

This language has a considerable syntactical complexity and requires advanced programming skills. Nevertheless, it has gained wide industrial attention, although no publications are yet known that use this notation for imaging applications. Because these skills are required for the implementation of image analysis as well, the language is useful for our purpose. We will use it in the following (e. g., in Fig. 5.3) and introduce the very basic notation.

Classes are represented as boxes, divided up to three fields; the upper field contains the class name,[1] the middle field contains data

[1]Instead of the technical name in the syntax of the programming language, we will use a descriptive term in the figures that follow.

fields called attributes, and the lower field contains method names. An extra small box in the left upper corner marks a template; the actual type is inserted here for a template instantiation. Except for the class name, the fields may be empty. Types and arguments are listed as well, as we will see in the examples to be given. Classes that are merely used as a common interface definition for further derived classes are called *abstract classes* in the object-oriented terminology; the name is printed in italics in UML.

An arrow relates two classes by inheritance, pointing to the base class. Template instantiations use a dashed line to relate to the template. A line with a filled diamond at its head denotes composition; an empty diamond is used for aggregation. In the following sections we will see several examples.

5.2.2 Object-oriented programming

After analysis and design, *object-oriented programming* (OOP) can take place. Classes are used for the implementation of actions or tasks, that is, algorithms, as well as information, that is, data. As we outline in Section 5.4.2, classes can also be used to provide easy and portable access to devices such as frame grabbers, or access to actors that are commonly used in active vision. As shown in Chapter 6, implementations for image operations can gain run–time efficiency if generic modules are used. Classes can be used to wrap these generic interfaces and to provide a uniform interface by inheritance.

Software reuse is highly desired due to the high costs of programming. Modularity is a central concept that helps maintain large systems. Data abstraction provides clean interfaces that are essential when several programmers share code in a team. One other goal of software engineering is to provide components with a long lifetime, even when changes are required. These principles have been known for years in the context of object-oriented programming; they have now gained wider attention. Object-oriented design cannot guarantee that these principles are fulfilled, but the strong interconnection of OOD, OOA, and OOP simplifies updates and evolution. In contrast to traditional software engineering, these three stages are not strictly sequential; a return from a later stage to a previous one is possible and intended.

The programming language C++ in particular has the advantage that it combines efficient conventional constructs with object-oriented features. Existing routines in C that sacrifice clean structure to gain speed—which unfortunately is necessary in some rare cases for image processing—can be encapsulated in classes or objects that provide safe interfaces.

Generally, methods in object-oriented programming have fewer arguments than corresponding function calls in traditional programming, because parts of the required information may already be bound to the object (e.g., an FFT object may have its internal tables for the actual image size, and no such tables will have to be allocated for the function call and passed to it as arguments). This again facilitates software maintenance and reuse, especially if a class and its interface have to be exchanged. Fewer modifications are then required in the code, compared to conventional programs.

Reuse of general *class libraries* serves two purposes. Common programming problems—like the implementation of linked lists or sets—have already been solved in these systems and can be used without further effort. Exchange of software using such class libraries is simplified because the classes share the same structure and interfaces.

The major idea of object-oriented programming now is to define abstract interfaces in general classes, that is, classes which are in the higher levels of the class inheritance graph, and to provide a specific implementation in the derived classes. If an algorithm now uses only the interfaces available in the more general classes, then the outcome of the process depends on the actual object to which the method is applied. The type of object may vary and can be taken from several derived classes. This behavior is called *polymorphism*.

5.2.3 Software engineering for image processing

Real-time constraints, efficiency, and the large amount of data impose special software problems to image processing systems. The basic requirements for designing a general software system for image processing (in the sense of the invariant part of a system in Section 4.1.1) are:

1. Access to imaging hardware has to be possible; this includes capturing devices, graphics, etc. as well as camera interfaces, camera motors, etc. Because hardware development occurs more rapidly than software can change, and because software reuse is desired, the interfaces have to be encapsulated by portable definitions.
2. Naturally, highly efficient—yet safe—access to vectors and matrices has to be possible.
3. Input and output has to be fast, efficient, and machine-independent. This has to be guaranteed not only for low-level data structures such as image matrices, but also for intermediate– and high-level data such as segmentation results and knowledge bases, as well.
4. Image processing and analysis modules should be as independent as possible from the final application, in order to be reusable across systems.

The discussion of object-oriented programming for image processing started when C++ became known. This programming language promised to provide the efficiency required for image processing, combined with object-oriented programming, and possible code reuse by upward compatibility to C.

The test phase is a crucial part of software design and programming. Systematically organized tests will increase the reliability of software for real-world applications. Problems not considered during the analysis might be detected during tests. A general problem is that tests can reveal only the existence of bugs, but generally cannot be used to prove that there are no errors in a program. The more you test, however, the lower will be the chance that there is a hidden bug in your code. Some *general guidelines* will help in testing your software:[2]

1. Put assertions in every method and function.[3] Test the module separately and undefine these assertions when either run-time is crucial for further tests, or when you are sure the assertions never fail.
2. Each branch in the program should be activated at least once during the test. This includes the parts for rare error conditions as well!
3. Every loop should be used at least twice during a test suite.
4. Try to imagine irregular, unexpected, wrong, inconsistent input and test your routines with such data, for example:
   (a) images of size $1 \times 10,000$ or even of size 0×0,
   (b) images with constant intensity value at every pixel,
   (c) sequences of zero length.
5. Use more than one image for testing. Vary all possible parameters, such as intensity, size, number of objects, contrast, etc.
6. Use at least one data set as input for which you know the expected output of the program.
7. Test predictable special cases, such as discontinuities of functions, division by numbers close to zero, etc.
8. Keep in mind the limitations of resources, such as storage or execution time. Estimate the required resources. Verify that predictable behavior of your program is guaranteed even if you exceed the limits.
9. Verify that users will accept your software and use it for their needs.

[2]Collected from the World Wide Web, from lecture notes, and from personal experience.

[3]Most C and C++ environments provide an efficient and effective solution by a simple `assert` macro.

5.3 Programming languages for image processing

We survey existing programming languages with respect to their usefulness for image processing and image analysis.

5.3.1 Conventional programming languages

Early image processing systems were mostly written in *Fortran*, for example, SPIDER [3]. Although this language still has its users, mostly because of its mathematical capabilities and libraries, very few image processing applications written in Fortran remain. After the ideas of IKS were dealt with in [4], several new software architectures for image processing were proposed, finally resulting in an international standard PIKS [5, 6, 7, 8]. This standard contains an interface to Fortran programs. A software architecture planned as a common basis for program exchange between companies, research institutes, and universities is presented in [9]. This system is now written in C++ and extends the ideas of SPIDER [3] and offers more than 500 algorithms for image processing.

For image processing, at least one decade was dominated by the use of C (see, e. g., [10, 11]). On the one hand, this language lacks most of the higher mathematical notation; on the other hand, it provides efficient access to low-level bit manipulations. The latter is very useful for low-level image manipulation, such as masking out areas in an image. The missing mathematical definitions in the language *syntax* are compensated by the large number of mathematical libraries available for the language. Rather than operators of the language, function calls to the libraries have to be used to apply mathematics.

The *Khoros* system [12, 13] is an environment for interactive development of image processing algorithms. The system includes a neat visual programming environment. The algorithms can also be run without interactive graphics. The system provides a large C-language library of imaging functions (over 500 functions), some of them used in 2 1/2-D and 3-D image processing. Knowledge-based processing is not part of this package. The PIKS standard mentioned here is also specified for the C language.

5.3.2 Object-oriented programming languages

Object-oriented programming languages have been known to computer scientists for more than 25 yr. The ideas originated in the ancestors Simula and Smalltalk. During this period of time, the (conventional) programming language C had its breakthrough.

In the late 1980s, the C language was extended with object-oriented ideas. The language C++ [14] mainly used the ideas of Simula. C++ is almost a superset of C; that is, most C programs are C++ programs as well. Many valuable image processing routines written in C can now be *reused* without modification. Possibly because of the inexpensive or free availability of C++ compilers—even on personal computers—this language had enormous success.

Several new object-oriented languages have been invented, such as Eiffel and CLOS, to name only two. In the following, we report only on those languages that are currently relevant in imaging applications.

The programming language C++ [14] has had overwhelming success in the last few years in all application areas. Many image processing projects started in C and are now extended or converted to C++, for example, [13, 15]. Although the C++-language is only partially suited for object-oriented programming, it seems to be the best choice to combine efficiency with object-oriented design in real-world applications, especially those operating under real-time conditions. For a discussion of image processing systems (see, e.g., [16, 17, 18, 19, 20]).

One major disadvantage of C++ was its lack of standard general purpose classes, such as linked lists. Meanwhile, the *Standard Template Library* (STL) [21] has become part of the C++ standard and is distributed with most compilers. Because it uses templates rather than inheritance, it gains run-time efficiency at some points compared to object-oriented class libraries. The STL supports genericity (Chapter 6); this is a somewhat orthogonal idea to object-oriented programming, which is based on inheritance. In C++, *polymorphism* is supported at run-time by `virtual` functions; these functions are internally called indirectly via a pointer, rather than by a direct function call. Template instantiation at compile-time provides a variant called *compile-time polymorphism*.

General purpose classes are also available for C++ as class libraries. A system called NIHCL [22] incorporates many Smalltalk ideas into C++ and uses inheritance rather than templates. In contrast to STL, NIHCL provides storage and recovery of arbitrary objects in a unified way, as was possible in Smalltalk. Our image analysis system described in Section 5.5 uses this general library.

For several years, there has been work on the image understanding environment; examples can be found in [23] and in various articles in the proceedings of the *Image Understanding Workshop*, for example, [24]. This large system is implemented in C++ as well and partially uses STL. The machine vision system described in [25] is implemented in C++ and uses class hierarchies for the knowledge base as well as for the interfaces to image processing algorithms.

The programming language *Java* [26] promises a new era in programming[4]. It is defined as a portable language where the portability extends down to the level of binary programs. This is achieved

[4]for example, `http://rsb.info.nih.gov/ij/`

by the so-called "Java virtual machine," which *interprets* the compiled programs. This interpreter has to be present on the host machine or hardware support has to be provided. Java is an object-oriented language with a clean and simple syntax definition, much simpler than C++ although with a similar outlook. The language is type safe, which helps create reliable programs. The language has no pointer variables, which is somewhat unusual. Many deficiencies of C++ have been cured in this new language. The compilers come with a rich library of general purpose classes that greatly simplifies sharing code with others. Although highly efficient implementations exist and applications to signal processing have been proposed [26], Java currently lacks the required run-time efficiency for image processing applications.

In its original definition [27], the programming language ADA was an object-based language; it was not an object-oriented programming language because it did not support inheritance, which is an essential object-oriented feature. Because of the high complexity of the formal syntax, it took several years before complete compilers were available on any platform. Although the language provides efficient, portable libraries, and concepts for software reuse, it was not accepted by the image processing population. Instead, the language C was used all over the world. The recent redefinition in [27] added the missing object-oriented features. The syntax complexity remained, however.

Smalltalk is *the* prototype of the object-oriented paradigm. As it is an interpreted language it is generally slower than a compiled program. It is also not particularly suited for mathematical problems. Smalltalk can be used to program the higher levels in image understanding; since the language has a large number of classes for various general problems, solutions can be formulated elegantly. When it comes down to pixel access for image preprocessing, the language is not the right choice.

5.3.3 Proprietary image processing programming languages

Some systems use an own programming language for image processing, which is usually either interpreted from a textual description or a graphical user interface. The *Cantata* language in *Khoros* discussed in [12, 13] is one typical example. *heurisko* [28] uses a textual as well as a graphical interface. *Ad Oculus* [29] has a graphical description. The major advantage is that usually only a few keystrokes are required for even complex imaging operations. The drawback is that a new language has to be learned and that this language is not portable to other systems, unless the interpreter exists on the target platform. One special case of this idea is image algebras where image processing is formalized as an algebraic problem (see [30]).

Table 5.1: *Features of object-oriented programming languages; extended from the table in [33]*

	Type-safe	Syntax complexity	ip/ia*	Concurrency tools
C++	low	high	yes	not part of the language
Java	yes	low	no	primitives exist
Ada	almost	high	few	fully supported
Smalltalk	yes	low	no	coroutines

* suited for ip—image processing; ia—image analysis

If image processing is seen as a data flow problem (Section 5.4.1), the command interpreter of the operating can be used to compose operations from single processes. This strategy has been applied in the system HIPS [31], which uses Unix-pipes to create sequences of operations. The Khoros system also provides a command line interface; the automatically created sequences of program invocations use intermediate files as interfaces. The system presented in [32] uses the Tcl/Tk language and interface to compose image understanding algorithms. The components may be written in C or C++.

5.3.4 Summary

In Section 5.2 we argued that object-oriented ideas greatly help reduce difficulties in software for image processing. In Table 5.1 we summarize those features of object-oriented programming languages that are of interest for imaging. Parallel processing is, of course, useful for image processing, especially when processing times are crucial. Although we did not treat this subject in the previous sections, we list in Table 5.1 whether concurrent and distributed processing is supported by the syntax of the programming language.

Although ADA'95 provides object-oriented features, it is barely used in scientific projects, except in those related to space technology; the programming languages C, C++, and Java are currently used in image processing implementations. In the table we note whether the language is currently used for image processing and image analysis (ip/ia). We also have a column for whether the language is type-safe; this feature, if provided, makes the tests performed by the compiler more reliable and thus increases software stability.

Although most programming languages allow mixtures with subroutines of other languages, we assume that the design goal in most cases is to have a uniform architecture as well as a uniform programming language. The C function calls in C++ can be seen as almost conformant with this guideline and are acceptable, because the benefit of reusing existing C code is much higher than the penalty of some minor extra interface requirements.

The language Objective-C encapsulated Smalltalk features in the C language. The system described in [16] is one of the few references to applications of Objective-C in imaging applications, combined with object-oriented extensions to LISP and Prolog. Today's best choice for a programming language for image processing and understanding is C++.

5.4 Image understanding

In this section we describe the general software architecture of *image understanding* (IU) systems and apply object-oriented principles.

5.4.1 Data flow

The general problem of image analysis is to find the best description of the input image data that is appropriate to the current problem. Sometimes this means that the most precise description has to be found, in other cases a less exact result that can be computed more quickly will be sufficient. This task may be divided into several subproblems. After an initial preprocessing stage, images are usually segmented into meaningful parts. Various segmentation algorithms create so-called segmentation objects [34]. Segmentation objects are matched with models in a knowledge base, which contains expectations of the possible scenes in the problem domain.

The various data types involved in image segmentation, like images, lines or regions, may serve for data abstraction of the problem. In object-oriented programming, these data types are naturally represented in classes. Segmentation may be seen as a data flow problem relating these various representations. An overview of the main components is shown in Fig. 5.1; data is captured and digitized from a camera and transformed to a symbolic description or results in an action of the system. Image processing tasks are shown in oval boxes; data is depicted as rectangles; in the object-oriented design phase, both will naturally be grouped to class hierarchies. If required, the algorithms can be implemented as generic modules for different types, as shown in Section 6.2.

The problem of finding an optimal match and the best segmentation can be seen as an optimization problem and is formulated as such in Volume 2, Chapter 27. Optimization may search for the best possible match as well as include efficiency considerations that are crucial for real-time image processing.

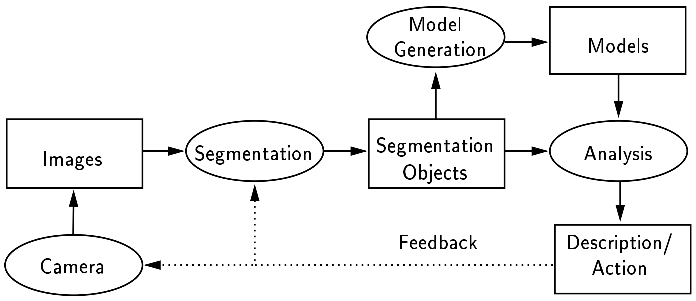


Figure 5.1: *Data flow in an image analysis system.*

Knowledge-based vision uses a model of the scene for image interpretation. The scene may be decomposed into object-models that can be represented using their structural properties and relations (e. g., in a semantic network Volume 2, Section 27.2.1), or as statistical object models ([35]; and Volume 2, Chapter 26).

The system architecture in Fig. 5.1 implies various interactions between modules. Information has to be exchanged by means of well-defined interfaces. Modules and data structures with well-defined interfaces are naturally implemented as classes in OOP or as modules in structured programming.

The segmentation object is a central idea for the data representation independent of the algorithms used for image segmentation, and can be used as an interface between data-driven segmentation and knowledge-based analysis. Models can be described in a similar formalism [36].

5.4.2 Devices and actors

The output of the system in Fig. 5.1 is a description of the input image data. In *active vision systems* (Chapter 9), the output may additionally or alternatively contain control commands for the sensor device or for the actor (e. g., a robot or a moving vehicle). This provides feedback from high-level processing to data-driven segmentation, or even to the image capturing devices (dashed line in Fig. 5.1). This data flow is also common to active vision systems, the parameters of the visual system are adjusted by the controlling computer in order to get the best possible image for the actual analysis purpose. The system design in Fig. 5.1 is, therefore, suitable for conventional image analysis as well as for active vision.

Interaction with graphical and pointing devices is necessary for interactive computer vision systems, as they are common in medicine, for example. Interaction with physically moving tools requires a control loop that can be closed in real-time.

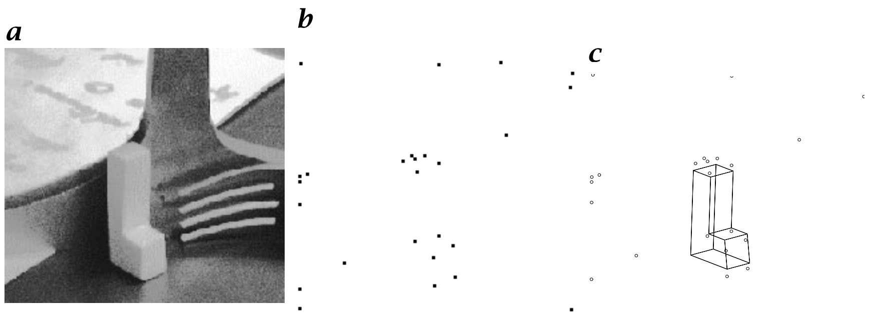


Figure 5.2: *Example of a scene with heterogeneous background:* ***a*** *gray-level image;* ***b*** *segmentation result; and* ***c*** *estimated pose*

Naturally, these actions and devices are modeled as classes and objects in OOA and OOD. Their well-defined interface facilitates data exchange between programs and devices.

Example 5.3: OOD for a frame grabber

Image processing software has to access imaging hardware, for example, to a *frame grabber*. Most systems provide libraries that are to some extent portable to the next generation of the same hardware. Although the functionality of two frame grabbers of different manufacturers may be rather similar, the software interface will of course be different.

Instead of the traditional approach, which scattered `#ifdef`'s around in the code to be able to compile a program for different interfaces, the object-oriented designer creates a class that encapsulates the common features of several hardware interfaces, for example, for setting the resolution of an input image or for selecting the input connector. The algorithms use this general—polymorphic—interface and are independent of the underlying hardware.

5.4.3 Statistical object recognition

In a Bayesian framework for *object recognition* using 2-D images ([37]; and Volume 2, Section 26.4), statistical model generation, classification, and localization is based on projected feature vectors. Localization is expressed by the rotation and projection matrix and translation vector. The objects are taken from a set of possible classes $\kappa \in \{1 \cdots K\}$ and described by parameter vectors, matrices, or sets.

Object localization and recognition corresponds to a general maximization problem (Volume 2, Eq. (26.4)).

Figure 5.2 shows a gray-level image (a) and the resulting set of 2-D point features (b), if a standard corner detection algorithm is applied.

Figure 5.2c shows the result of the maximization material dealt with in Volume 2, Eq. (26.4). Similar results are shown in Volume 2, Fig. 26.7.

In order to implement such an optimization, the function to be optimized has to be independently specified from the algorithm performing the optimization. Whereas this is written down easily in mathematics, it requires clean design in the implementation, taking into account the computational complexity of the optimization. We outline a solution in Section 5.5.3.

5.5 Class hierarchy for data and algorithms

Whereas hierarchical data representation by classes has become almost state of the art, hierarchies of classes for operations, algorithms, and actors are not yet common.

We first describe a *hierarchy of classes* that facilitate simple interfaces for image processing; image processing operations are implemented as another class hierarchy that uses the data interfaces [18]. A third hierarchy provides various optimization algorithms. These ideas are realized in **An** **im**age **a**na**l**ysis **s**ystem (ANIMALS), which is written in C++.

5.5.1 Data representation

Various data representation schemes have been developed for data in image analysis covering image data structures as well as results from segmentation. Some of them may be treated as algebras with more or less complete sets of operations (e. g., chain codes or quad trees). Some representations are used because of their storage efficiency (e. g., run length codes), others because of their runtime efficiency. Such ideas were combined into a **Hi**erarchy of **P**icture **P**rocessing **O**bject**S** (HIPPOS, written as ἵππος [34]).

A central problem visible in Fig. 5.1 is the data exchange of segmentation results, which shows an initial symbolic description of the image. The solution in ἵππος is a hierarchy of classes, which is shown in Fig. 5.3. The general classes for implementation, which are added in OOA (Section 5.2.1) were taken from the NIHCL class library [22]. All segmentation results can be stored in an object of class `SegObj` no matter whether the data is computed from line- or region-based segmentation, color, range, or gray-level images. This object representation can be exchanged between different processes that could run on different computer architectures. This general tool is used in many algorithms. A segmentation object is a geometric object (`GeoObj`). It consists of a set of parts, which are in turn geometric objects. Atomic objects are geometric objects as well and end the recursion. The abstract base class

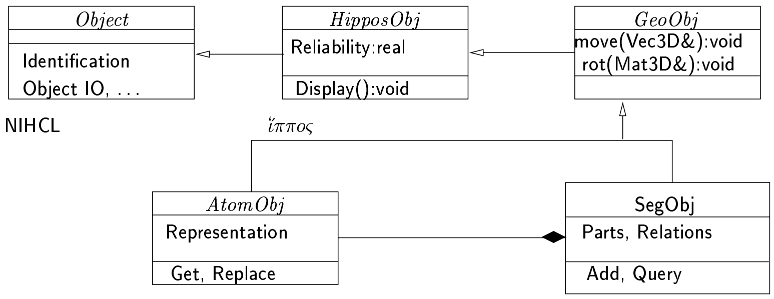


Figure 5.3: *A section of a hierarchy of geometric objects for segmentation (in total approximately 150 classes).*

`HipposObj` connects to the NIHCL object and serves as a common root for all image processing classes. It contains a judgment attribute that is used for the comparison of competing segmentation results by the analysis control (compare Volume 2, Section 27.6).

As our applications, for example, in Denzler and Paulus [38] and Beß et al. [39] show, this has proven adequate for 2-D, 2 1/2-D and 3-D image analysis. Other subtrees exist for image classes, like gray-level images, stereo images, range images, color images, etc. as well as classes for lines, circular arcs, polygons, chain codes, regions, active contours etc. The whole *ἵππος*-hierarchy currently consists of approximately 150 classes.

Vectors and matrices are implemented as templates using the concept of genericity (Chapter 6). In order to be efficient, pixel access to images should not be programmed via virtual functions. A flexible, safe and simple mechanism is used instead, which is shown in Example 5.4. Pixel access by operator syntax is possible for vectors as well as for matrices without loss of speed [18]. The index operator in Example 5.4 can do a check on the validity of the operand, thereby eliminating most of the common errors in image processing (which cannot be detected in C automatically).

Example 5.4: Efficient and safe pixel access

```
template <class T> struct Vector { // simplified version
  T * array; int size;             // the actual data
public:
  T& operator[] (int i)
{ /* check index validity here */ return array[i]; }
  Vector(int);               // allocate internal pointer
  Vector(int,T* p);          // use already existent storage in p
};
template <class T> struct Matrix { // simplified version
  T ** tarray; int size;    // contiguous allocation for the matrix
  Vector<T>** varray;        // the varrays internally use tarray
public:
  Matrix(int,int);        // will allocate C-compatible tarray and
  // use second constructor for vectors
  // using this array
  operator T**() { return tarray; }  // provides C compatibility
  Vector<T>& operator[] (int i)
{ /* check index here */ return *varray[i]; }
};

void t()
{
   Matrix<int> m(10,10);  // define a matrix object
   m[3][4] = 3;           // safe pixel access
   int** mp = m;          // convert to pointer access
   mp[3][4] = 4;          // fast pixel access
}
```

The idea in Example 5.4 can be used to implement subimages [18]. An interface to commonly available C-functions is easily done by the automatic conversion operator to type T**; because these generic modules are integrated into the NIHCL class hierarchy, they share methods for input and output of objects.

5.5.2 Image operator hierarchy

Many operations in image processing can be structured hierarchically in a straightforward manner. Such hierarchies can be implemented in a hierarchy of classes for *operations* in a straightforward way (see [16, 40]). Objects are the actual algorithms with specific parameter sets, which are also objects [41]. Classes as implementation of algorithms are particularly useful, when operations require internal tables that increase their efficiency. The requirement stated in Section 4.2 that algorithms should provide meta-knowledge about themselves can be fulfilled by operator classes; provided polymorphic methods list the type of operator, the required arguments, and further administrative information. Programmers can be forced by the compiler to implement these functions.

The major advantages of operator-classes are 3-fold.

- Algorithms can be programmed in an abstract level referencing only the general class of operations to be performed; extensions of the system by a new derived special operator will not require changes in the abstract algorithm.
- Such an extension cannot change the interface, which is fixed by the definition in the abstract base class. This guarantees uniform and thus easy-to-use interfaces.
- Dynamic information about the operator, which is actually used, is available. For example, a program may just reference a filter object; during run time it will be decided which concrete filter should

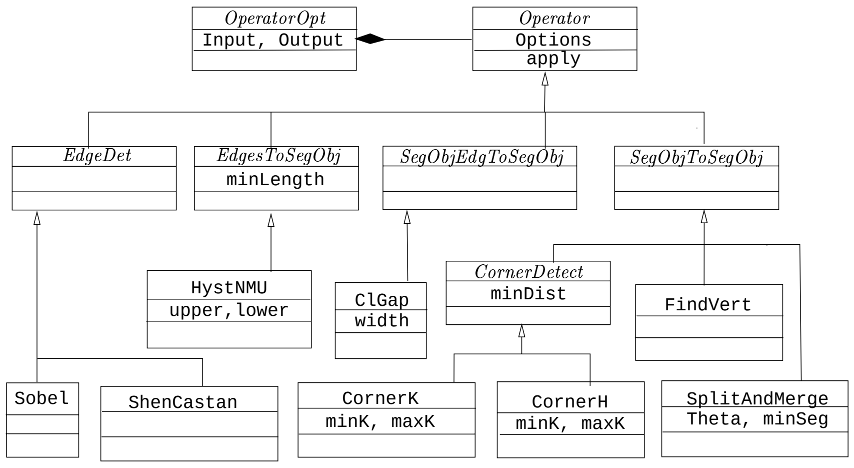


***Figure 5.4:** Subtree of the operator hierarchy for image segmentation; from [41].*

be used. Using virtual functions in C++, the run time overhead is negligible (Section 5.5.5).

The segmentation in [41] accepts as input for an image object. An edge-operator-object such as a Sobel-object or a Nevatia-object converts this input image to an edge image. Edges are linked to line objects collected in a segmentation object. Corners are detected on the line and added to the segmentation object with one of several corner detector objects. Later, vertices are identified and the lines are approximated by circular arcs and straight lines using a split-and-merge object. This sequence of operators introduces a refinement of the data flow in Fig. 5.1. Figure 5.4 shows the hierarchy for line segmentation. The segmentation result in Fig. 5.2 as well as the results in Volume 2, Fig. 26.7 are computed using this sequence; both point sets are collected in an object of class `SegObj`. Parameter blocks can be shared between different operators as parameter objects. Function call C++ syntax for these operators as shown in Example 5.5 is used in Animals, which facilitates migration of existing conventional programs to operator objects.

Example 5.5: Image operator classes in C++

```
class Sobel: public EdgeDet { // Sobel operator as a special case
public:                       // of an edge detector
   static const int maxStrength;  //  = 2040;
   virtual void operator() (const GrayLevelImage&,EdgeImage&) ;
};
class Prewitt : public EdgeDet {  // Prewitt edge operator
public:
   static const int maxStrength;  //  = 1020;
```

```
    virtual void operator() (const GrayLevelImage&,EdgeImage&) ;
};
```

Whereas *one* object of a Sobel object will usually be sufficient in a program, several objects of a certain operator object will be needed if these objects keep their internal state and auxiliary data between invocation. As an example consider a discrete Fourier transform that uses tables for sine and cosine values; the size and contents of these tables vary, however, upon the frame size to be transformed. Several Fourier-transform objects may thus be used in one program for transformations of images, for example, in a resolution hierarchy. In contrast to conventional solutions, this requires neither code duplication nor complicated management of functions with local storage. In Example 5.6, a simple interface to an FFT object is shown. Each instance of the FFT class has an associated internal size and can be used to transform a vector of this size. Again, function call syntax is used.

Example 5.6: Implementation of class FFT

```
class FFT : public IP_OP {
  Vector<double> sintab, costab;  // internal tables
public:
  FFT(int s) : sintab(s), costab(s) { /* init tables here */ }
  Vector<complex> operator() (const Vector<double>&); // apply FFT
};

void f()
{
  FFT f128(128);  // init internal tab for 128 entries
  FFT f256(256);  // init internal tab for 256 entries

  Vector<double>  v128(128), v256(256); // input data
  Vector<complex> c128 = f128(v128);    // resulting spectrum 1
  Vector<complex> c256 = f256(v256);    // resulting spectrum 2
}
```

5.5.3 Hierarchy for optimization algorithms

The optimization problem in Volume 2, Eq. (26.4) and in Volume 2, Example 26.6 requires that several strategies for optimization be evaluated in order to find efficient object recognition strategies. Probabilistic optimization routines that allow practically efficient solutions are discussed in [37]. Again, a class hierarchy for optimization strategies similar to the operator hierarchy already mentioned here simplifies the experiments.

The basic idea of the implementation is to program the algorithms independently from the function to be optimized. An abstract base for all optimization strategies has an internal variable, which is the function to be optimized; the class provides a method for minimization or maximization to all derived classes.

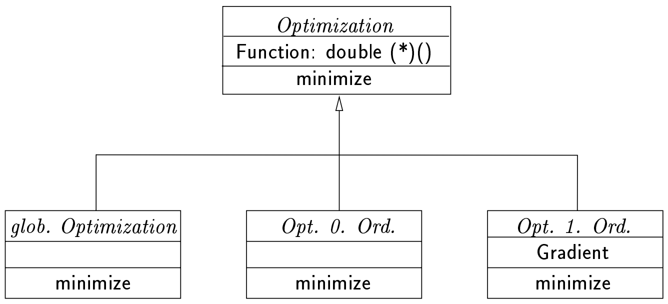


Figure 5.5: *Partial view on class hierarchy for optimization algorithms.*

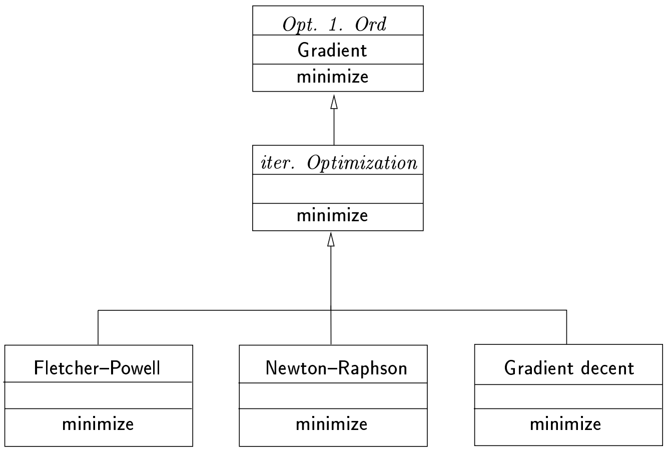


Figure 5.6: *Partial view on class hierarchy for local optimization algorithms of first order.*

All optimization algorithms can be divided into global and local procedures; additional information may be present, such as, for example, the gradient of the function (Fig. 5.5). Procedures that use the function directly are the combinatorial optimization, the simplex algorithm, or the continuous variant of the simulated annealing algorithm (Fig. 5.7). The gradient vector can be used for the optimization of first order. Examples are the iterative algorithms implemented in [37], the algorithm of Fletcher and Powell, and the well-known Newton-Raphson iteration, all of which are shown in Fig. 5.6. Figure 5.8 finally shows an overview of the other algorithms implemented in [37].

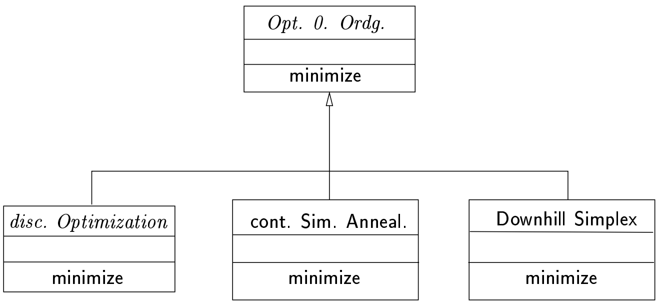


Figure 5.7: *Partial view on class hierarchy for model densities.*

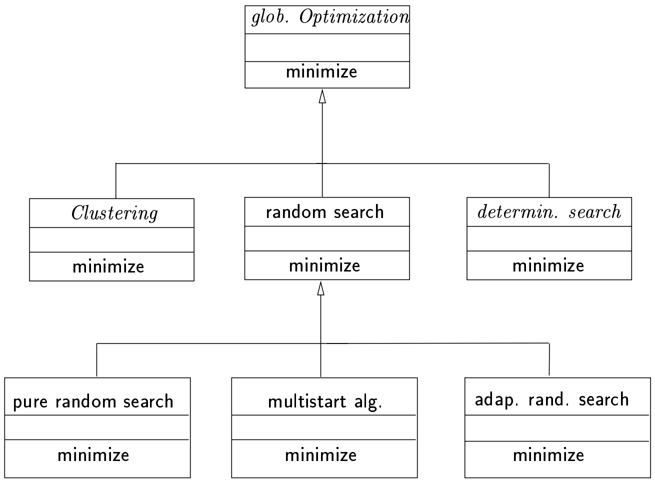


Figure 5.8: *Partial view on class hierarchy for global optimization algorithms.*

5.5.4 Hierarchy for actors and devices

A technique similar to that in Example 5.5 can be used to fulfill requirement 1 on page 82. This is shown in Example 5.7 for the case of a camera that can capture either gray-level images or color images and a pan/tilt unit that is commonly used in active vision systems. Of course, a more elaborate internal structure of the `class SMotor` is required in real applications. In fact, a hierarchy of classes providing different interfaces to various hardware is used. This is invisible for the application programmer.

Example 5.7: Actor classes in C++

```
struct SMotor {      // assume stepper motor
  operator=(int p); // will set motor to position p
```

Table 5.2: *Run-time measurements for HP (735/99 MHz), SGI O2 (R10000, 195 MHz), and Dual-Pentium (Linux 300 MHz); all times in milliseconds*

	Number of calls	HP	O2	PC
Direct function call	10^7	199	93	43
Direct method call	10^7	233	93	55
Virtual method call	10^7	634	98	65
Sobel function 256^2 image	10^2	1085	287	300
Sobel object 256^2 image	10^2	1096	290	301
Sobel function 32^2 image	10^4	1551	418	415
Sobel object 32^2 image	10^4	1557	419	175
Safe pixel access	10^7	446	246	84
Fast pixel access	10^7	162	72	37

```
  operator++();     // will move by one step
};
struct PTVDev : public IP_DEV {
  SMotor panaxis;
  SMotor tiltaxis;
};
struct Camera : public IP_DEV {
  SMotor zoom;                       // zoom motor is stepper motor
  virtual void operator() (GrayLevelImage&); // captures correct
  virtual void operator() (ColorImage&);     // type and size
};
```

Most common functions of such devices can be accessed by operator-syntax. This releases the programmer from even the need of remembering method names because the value for the stepper motor can simply be assigned to the object. The challenging idea behind this concept is not only to encapsulate the hardware interface for each device, but also to provide common interfaces for parts of the device, such as motors.

5.5.5 Efficiency

As shown in Table 5.2, time *overhead* for virtual function calls is negligible compared to the time needed for the actual computation, such as in the case of the Sobel operator. Of course, trivial computations or pixel access should not be performed using virtual functions, as the ratio of overhead and required processing time for the task is then worse. The implementation of the Sobel operator uses the fast pixel access via pointers (Table 5.2).

In [42], a *real-time active vision system* was implemented for tracking objects and moving a pan/tilt camera. This system used the image and matrix classes as well as the camera and actor classes in Section 5.5.

The computation times were small enough to process images and control the camera.

The computation times measured in [37] for the optimization classes outlined in Section 5.5.3 were around one minute for 10,000 calls to the function to be optimized.

The system has been used in many other applications as well. These references and the measures in Table 5.2 prove that object-oriented programming can be successfully applied in image processing: low-level tasks on matrices and vectors use templates; intermediate results and high-level processing rely on class hierarchies.

The benefit of using object-oriented programming instead of conventional programming is hard to measure. One measure for the quality of a software system is the duration of its use, which is long if the architecture is accepted by the implementors and which is shorter if the system fails to meet the requirements of the programmers. Our system was started in 1988 and the oldest class still in use is the one for chain codes (which was released in 1989). Our personal experience strongly suggests object-oriented principles for system design in image analysis.

5.5.6 Summary

In Section 5.2.3 we listed three essentials for imaging software design. In Section 5.5.5 we showed how efficient processing is possible simultaneously with object-oriented design. Several examples were given in the previous sections on access to hardware and devices. The NIHCL object, which is used as the base class for all classes shown in the preceding text, provides machine-independent persistent objects. This data representation is available not only for images, segmentation objects, etc., but also for actors such as a camera; in this case, the camera parameter settings are stored and can easily be retrieved later.

Of course, object-oriented programming implies that the complete class hierarchy has to be available for the application programmer, for example, as a shared library. In contrast, the independent building blocks proposed in Chapter 6 require almost no common libraries, but share the code for the generic modules. A combination of both approaches will give the best results for complete imaging systems.

5.6 Conclusion

We argued for a clean and uniform software design in imaging software systems. At the same time, a highly efficient implementation is crucial. Only C++ can currently provide these features. For regular data structures such as matrices and vectors, genericity is the right choice; in C++ this concept is available as templates. Starting with images that use such matrices, and ending with complex data representation as segmentation results, classes and inheritance provide clean yet efficient interfaces.

In addition to data, algorithms are grouped hierarchically as classes. While not imposing measurable overhead on the run-time, this idea is significant in keeping interface problems small and forcing programmers to adhere to standards. The use of classes for interfaces to hardware is to guarantee platform independent access. Hierarchies of such classes facilitate uniform syntax even for varying devices. A wise choice of the available tools will keep the run-time overhead at a negligible minimum and will at the same time greatly increase the value of the software in terms of reusability and expected maintenance effort.

Acknowledgment

The authors want to thank J. Meixner for his remarks on Section 5.2. We acknowledge the software contribution of our various colleagues. Numerous students helped to implement the system.

5.7 References

[1] Cardelli, L. and Wegner, P., (1985). On understanding types, data abstraction, and polymorphism. *Computer Surveys*, **17**(4):471–522.

[2] Breu, R., Hinkel, U., Hofmann, C., Klein, C., Paech, B., Rumpe, B., and Thurner, V., (1997). Towards a formalization of the unified modeling language. In *ECOOP'97—Object-Oriented Programming, 11th European Conference, Jyväskylä, Finland*, M. Aksit and S. Matsuoka, eds., Vol. 1241 of *Lecture Notes in Computer Science*, pp. 344–366. Berlin: Springer.

[3] Tamura, H., (1983). Design and implementation of SPIDER—a transportable image processing software package. *Computer Vision, Graphics and Image Processing*, **23**:273–294.

[4] Gemmar, P. and Hofele, G., (1990). An object-oriented approach for an iconic kernel system IKS. In *Proceedings of the 10th International Conference on Pattern Recognition (ICPR)*, Vol. 2, pp. 85–90. Atlantic City: IEEE Computer Society Press.

[5] Blum, C., Hofmann, G. R., and Kromker, D., (1991). Requirements for the first international imaging standard. *IEEE Computer Graphics and Applications*, **11**(2):61–70.

[6] Butler, T. and Krolak, P., (1991). An overview of the Programmer's Imaging Kernel (PIK) proposed standard. *Computers and Graphics*, **15**(4):465–472.

[7] ISO, (1994). *International Standard 12087, Image Processing and Interchange*. Technical Report, International Standards Organization, Geneva.

[8] Pratt, W. K., (1995). *The PIKS Foundation C Programmers Guide*. Greenwich: Manning.

[9] Kawai, T., Okazaki, H., Tanaka, K., and Tamura, H., (1992). VIEW-Station software and its graphical user interface. In *SPIE Proc. Image Processing*

and Interchange: Implementation and Systems, R. B. Arps and W. K. Pratt, eds., Vol. 1659, pp. 311–323. San Jose, CA: SPIE.

[10] Dobie, M. R. and Lewis, P. H., (1991). Data structures for image processing in C. *Pattern Recognition Letters*, **12**:457–466.

[11] Parker, J. R., (1997). *Algorithms for Image Processing and Computer Vision.* New York: Wiley Computer Publishing.

[12] Rasure, J. R. and Young, M., (1992). Open environment for image processing and software development. In *SPIE Proc. Image Processing and Interchange: Implementation and Systems*, R. B. Arps and W. K. Pratt, eds., Vol. 1659, pp. 300–310. San Jose, CA: SPIE.

[13] Young, M., Argiro, D., and Kubica, S., (1995). Cantata: Visual programming environment for the Khoros system. *Computer Graphics*, **29**(2):22–24.

[14] Stroustrup, B., (1991). *The C++ Programming Language*, 2nd edition. Reading, MA: Addison-Wesley.

[15] Eckstein, W., Lohmann, G., Meyer–Gruhl, U., Riemer, R., Robler, L. A., and Wunderwald, J., (1993). Benutzerfreundliche Bildanalyse mit $\mathcal{HORUS}$: Architektur und Konzepte. In *Mustererkennung 1993*, S. J. Pöppl and H. Handels, eds., pp. 332–339. Springer.

[16] Carlsen, I. C. and Haaks, D., (1991). IKS^{PFH}—concept and implementation of an object-oriented framework for image processing. *Computers and Graphics*, **15**(4):473–482.

[17] Coggins, J. M., (9.-10. November 1987). Integrated class structures for image pattern recognition and computer graphics. In *Proceedings of the USENIX C++ Workshop*, K. Gorlen, ed., pp. 240–245. Santa Fe, NM.

[18] Paulus, D. and Hornegger, J., (1997). *Pattern Recognition of Images and Speech in C++*. Advanced Studies in Computer Science. Braunschweig: Vieweg.

[19] Piper, J. and Rutovitz, D., (1988). An investigation of object-oriented programming as the basis for an image processing and analysis system. In *Proceedings of the 9th International Conference on Pattern Recognition (ICPR), Rome*, Vol. 2, pp. 1015–1019. Washington, DC: IEEE Computer Society Press.

[20] Zimmer, W. and Bonz, E., (1996). *Objektorientierte Bildverarbeitung*. München: Carl Hanser Verlag.

[21] Musser, D. R. and Saini, A., (1996). *STL Tutorial and Reference Guide*. Reading, MA: Addison-Wesley.

[22] Gorlen, K. E., Orlow, S., and Plexico, P. S., (1990). *Data Abstraction and Object-Oriented Programming in C++*. Chichester: John Wiley and Sons.

[23] Haralick, R. M. and Ramesh, V., (1992). Image understanding environment. In *SPIE Proc. Image Processing and Interchange: Implementation and Systems*, R. B. Arps and W. K. Pratt, eds., Vol. 1659, pp. 159–167. San Jose, CA: SPIE.

[24] Mundy, J., Binford, T., Boult, T., Hanson, A., Veveridge, R., Haralick, R., Ramesh, V., Kohl, C., Lawton, D., Morgan, D., Price, K., and Strat, T., (1992). The image understanding environments program. In *Image Understanding Workshop*, pp. 185–214. San Diego, CA.

[25] Caelli, T. and Bischof, W., (1997). Machine learning and image interpretation. In *Machine Learning and Image Interpretation*. New York: Plenum.

[26] Lyon, D. A. and Rao, H. V., (1997). *Java Digital Signal Processing*. Redwood City, CA: M&T Books.

[27] Barnes, J., Brosgol, B., et al., (1995). *Ada 95 Rationale. The Language. The Standard Libraries*. Technical Report, Itemetrics Inc., 733 Concord Av., Cambridge, MA 02138.

[28] Jähne, B., (1997). *Digital Image Processing*. Berlin: Springer.

[29] Bässmann, H. and Besslich, P., (1995). Ad oculos. In *Ad Oculos*. London: Internat. Thomson Publ.

[30] Giardina, C., (1984). The universal imaging algebra. *Pattern Recognition Letters*, **2**:165–172.

[31] Landy, M., (1993). HIPS-2 software for image processing: goals and directions. *SPIE*, **1964**:382–391.

[32] Cracknell, C., Downton, A. C., and Du, L., (1997). TABS, a new software framework for image processing, analysis, and understanding. In *Proc. Image Processing and its Applications*, pp. 366–370.

[33] Goedicke, M., (1997). Java in der Programmierausbildung: Konzept und erste Erfahrungen. *Informatik Spektrum*, **20**(6):357–363.

[34] Paulus, D. and Niemann, H., (1992). Iconic-Symbolic Interfaces. In *SPIE Proc. Image Processing and Interchange: Implementation and Systems*, R. B. Arps and W. K. Pratt, eds., Vol. 1659, pp. 204–214. San Jose, CA: SPIE.

[35] Hornegger, J. and Niemann, H., (1994). A Bayesian approach to learn and classify 3-D objects from intensity images. In *Proceedings of the* 12^{th} *International Conference on Pattern Recognition (ICPR), Jerusalem*, pp. 557–559. IEEE Computer Society Press.

[36] Niemann, H., (1990). *Pattern Analysis and Understanding*, Vol. 4 of *Springer Series in Information Sciences*. Heidelberg: Springer.

[37] Hornegger, J., (1996). *Statistische Modellierung, Klassifikation und Lokalisation von Objekten*. Aachen: Shaker Verlag.

[38] Denzler, J. and Paulus, D., (1994). Active Motion Detection and Object Tracking. In *First International Conference on Image Processing, Austin, Texas, USA*.

[39] Beß, R., Paulus, D., and Niemann, H., (1996). 3D recovery using calibrated active camera. In *Proceedings of the International Conference on Image Processing (ICIP), Lausanne, Schweiz*, Vol. 2, pp. 635–639. Washington, DC: IEEE Computer Society Press.

[40] Faasch, H., (1987). *Konzeption und Implementation einer objektorientierten Experimentierumgebung für die Bildfolgenauswertung in ADA*. PhD thesis, Universität Hamburg, Hamburg.

[41] Harbeck, M., (1996). *Objektorientierte linienbasierte Segmentierung von Bildern*. Aachen: Shaker Verlag.

[42] Denzler, D., (1997). *Aktives Sehen zur Echtzeitobjektverfolgung*. Aachen: Infix.

6 Reusable Software in Computer Vision

Ullrich Köthe

Fraunhofer-Institut für Graphische Datenverarbeitung
Rostock, Germany

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

6.1 Introduction

Since the mid-1980s general purpose computers such as workstations and PCs have reached performance levels that make them increasingly interesting for computer vision applications. As compared with applications based on special hardware, software solutions have a number of important advantages: they are easily transferred to the next hardware generation, can be adapted quickly to new requirements, and can be reused in many different contexts and systems.

It is, however, not easy to reach these goals in practice. The current praxis of software development in computer vision has not led to software that is as flexible and reusable as we would like it to be. The author's belief is that these problems are due to two fundamental difficulties, which will be addressed in this chapter:

- **The flexibility vs performance problem**
  In many cases, flexibility introduces a run-time overhead that degrades performance. Given the huge amount of data that needs to be processed in most computer vision applications, there is only a very limited leeway before performance is traded for flexibility. Thus, we must look for flexibility techniques that do not affect performance.
- **The framework coupling problem**
  Most reusable components are currently provided as parts of large frameworks. Because individual components in these frameworks are tightly coupled together, it is not possible to take out just those pieces we actually need: the framework functions only as a whole. This essentially prevents the use of components from several frameworks in the same project. Thus, we must provide reusable software in the form of independent building blocks that can be reused individually.

Let us illustrate these problems with an example. Suppose we have built an image processing application that we would like to adapt to the special needs of an important customer. The existing body of functionality is large, so that we can not afford to throw it away and start over from scratch. To fill in the missing pieces quickly, we would like to reuse some functionality that has been developed in another context, perhaps another project or the public domain.

For example, suppose we want to incorporate the Shen-Castan edge detector that is part of the KHOROS system [1] into our environment. Unfortunately, this algorithm only runs on a specific variety of the KHOROS image format. If we do not happen to use this image format in our system, we have an adaptation problem. This problem is traditionally solved in several ways:

- We can convert our original image representation into the required KHOROS format prior to every application of the algorithm, and convert the results back upon completion. This is the solution applied within KHOROS itself and within the *component-based* software paradigm in general (see Chapter 4). Clearly, conversion is relatively inefficient, because it takes time and memory. Thus, we would like to avoid it at the lower level of a system (although it might be a good option at higher levels).
- To minimize conversion, we can opt to build an entire module using KHOROS functionality: that is, besides the edge detector we also use KHOROS functions for the necessary pre- and post-processing. Although it might work in some cases, it is not a good idea to use several different frameworks within the same project—it leads to explosive growth in code size, and requires understanding and maintaining all these different frameworks, which might easily vitiate the positive effect of software reuse.
- If we have access to the source code (which is the case with KHOROS functionality), we can also attempt to modify the code to adapt it to the needs of our environment (e.g., let it run on top of our image format). This approach is very problematic as it easily introduces subtle bugs that are very hard to detect. To avoid mistakes, a very thorough understanding of the existing code is required, which many people find even harder than reimplementing the solution from scratch.

Neither of these options is really satisfying. It would be much better if reusable components were provided as *independent building blocks* that do not depend on a specific environment (such as KHOROS) and can be adapted without knowledge of their inner workings (in particular, without source code modification) and without loss of performance. The design methods described in previous chapters (Chapters 4 and 5) do not support independent building blocks very well, especially at the level of basic algorithms and data structures, where performance is at a premium. Therefore, in this chapter we describe an alternative design approach called “generic programming,” which provides very good support for definition and flexible integration of independent building blocks. Generic programming complements the other methods so that we can design combined solutions (such as object-oriented frameworks delegating their implementation to generic components) that emphasize the strengths of each method.

The most important dependency we need to remove is the dependency of algorithms upon specific data representations. To a certain extent, object-oriented programming provides a solution to this problem if data is accessed via abstract classes and virtual functions. An abstract image base class could, for example, look like this (all code examples are in C++):

```
class AbstractImage
{
  public:
    virtual unsigned char getPixel(int x, int y) const = 0;
    virtual void setPixel(unsigned char value, int x, int y) = 0;
    virtual int width() const = 0;
    virtual int height() const = 0;
    // ...
};
```

Algorithms using this abstract base class can not know what is behind the virtual functions. Thus, we can wrap arbitrary image formats into subclasses of this abstract image, for example the KHOROS xv image format:

```
class KhorosImage
: public virtual AbstractImage
{
  public:
    virtual unsigned char getPixel(int x, int y) const
    {
        return image->imagedata[x + y*image->row_size];
    }
    virtual void setPixel(unsigned char v, int x, int y)
    {
        image->imagedata[x + image->row_size*y] = v;
    }
    virtual int width() const { return image->row_size; }
    virtual int height() const { return image->col_size; }
    // ...
  private:
    xvimage * image;
};
```

However, there are two fundamental problems with this approach. First, the virtual function call introduces a run-time overhead. In Section 6.6 we report benchmark results that show a performance penalty of up to 500 % when we use virtual functions instead of accessing the data directly via pointers[1]. Of course, the overhead could be avoided if we implemented algorithms as member functions of the concrete image classes, giving them direct access to the data. But this would counter our desire for reusability, as these member functions would once again depend on the concrete image format and would not be independently reusable.

The second, more severe problem is the return type of the function `getPixel()`. In a statically typed language, this return type must be fixed (to `'unsigned char'` in the example) so that `'float'` and RGB images can not be subclasses of `AbstractImage`. This could be overcome by introducing another abstract class `AbstractPixel`, but this would require even more virtual calls so that it is not a practical pos-

[1]The overhead results primarily from the inability of the compiler to inline virtual functions. See Chapter 5 for further discussions.

sibility. Thus, for different return types of `getPixel()` we still need different versions of each algorithm.

Although organizing objects into class hierarchies has advantages (see Chapter 5), in a statically typed language it also introduces static dependencies, that is, all types (base classes, function arguments, and data members) and member functions that are mentioned in a class must also be transferred into the new environment if we want to reuse this class. This often leads to a chain of interdependencies which, is the main cause for the framework coupling problem mentioned in the preceding, and prevents the definition of really *independent* building blocks.

Therefore, an additional design technique is needed. Traditionally, the problems mentioned have been addressed by *code generators*. Code generators produce the source code for a concrete application from an abstract implementation of a building block and a description of the context (containing, among other things, the concrete data types involved). Meanwhile, these code generating facilities have been built directly into major object-oriented languages (such as C++, ADA, and Eiffel) under the name of *genericity* and *parametric types*. The most extensive support for genericity is provided by the *template mechanism* in C++. The widespread support of genericity enabled the development of a new programming paradigm—*generic programming*—which generalizes and formalizes the ideas of code generation and thus solves the forementioned problems without introducing a major run-time overhead. In this chapter we will describe how generic programming can be used to design reusable, independent building blocks for computer vision applications.

6.2 Generic programming

Generic programming was introduced as an independent programming paradigm by Musser and Stepanov [2, 3]. It became popular when it was chosen as the underlying paradigm of the C++ Standard Library, in particular the part of the library known as the *standard template library* [4]. It is especially suited to implement reusable data structures and algorithms at the lower levels of a system. Because generic programming is still relatively new, we will give a short general introduction before we start applying it to computer vision.

6.2.1 Generic data structures

In the first step of generic programming, container data structures are made independent of the type of the contained objects. In computer vision, we will implement *generic image data structures* that can contain arbitrary pixel types like this:

```
template <typename PIXELTYPE>
class GenericImage
{
  public:
    PIXELTYPE getPixel(int x, int y) const {
        return data_[x + width()*y];
    }

    void setPixel(PIXELTYPE value, int x, int y) {
        data_[x + width()*y] = value;
    }

    int width() const { return width_; }
    int height() const { return height_; }
    // etc.

  private:
    PIXELTYPE * data_;
    int width_, height_;
};
```

Alternatively, we can also use an additional array of pointers to the start of each line to enable double indirection, which is faster on many machines:

```
template <typename PIXELTYPE>
class GenericImageAlt
{
  public:
    PIXELTYPE getPixel(int x, int y) const {
        return lines_[y][x];
    }

    // etc.

  private:
    PIXELTYPE *  data_;
    PIXELTYPE ** lines_;
    int width_, height_;
};
```

In either case, the generic type `PIXELTYPE` acts as a placeholder for any pixel type we might want to store, and the corresponding image data structure will be automatically generated by the compiler when needed. For example, to create an RGB image, we could implement an RGB compound, and instantiate a `GenericImage` with it:

```
template <typename COMPONENTTYPE>
struct RGBStruct
{
    RGBStruct(COMPONENTTYPE r, COMPONENTTYPE g, COMPONENTTYPE b)
    : red(r), green(g), blue(b)
    {}

    COMPONENTTYPE red, green, blue;
};

// generate an new image data type holding RGBStruct<unsigned char>
typedef GenericImage<RGBStruct<unsigned char> > RGBStructImage;
```

6.2.2 Generic algorithms

The basic techniques to define generic data structures are well known and already widely applied. In the next step of generic programming we define algorithms as templates as well. For example, to copy arbitrary images (including the `RGBStructImage` already defined herein) we could write:

```
template <typename SRCIMAGE, typename DESTIMAGE>
void copyImage(SRCIMAGE const & src, DESTIMAGE & dest)
{
    for(int y=0; y<src.height(); ++y)
      for(int x=0; x<src.width(); ++x)
        dest.setPixel(src.getPixel(x, y), x, y);
}
```

This algorithm can be applied to any image pair for which a conversion from the source to the destination pixel type exists, provided the source image supports operations `src.height()`, `src.width()`, and `src.getPixel(x,y)`, and the destination image `dest.setPixel(v,x,y)`. We will call requirements like these an algorithm's *required interface* because they reflect what the user of the algorithm must ensure to allow the compiler to generate the code for a given pair of images.

Two properties of generic algorithms should be noted here: first, instantiation of the algorithm is completely type based. That is, any image that fulfills the required interface can be used within the algorithm—no inheritance from a specific abstract base class is required. Second, all decisions regarding flexibility are done at compile time (so-called *compile-time polymorphism*). Thus, when all access functions are expanded inline, the compiler can generate code that runs as fast as a traditional pointer-based (non-reusable) implementation[2].

6.2.3 Iterators

In practice, it turned out that algorithms with required interfaces similar to the previous one (i. e., algorithms that directly access the data structures' member functions) are not the optimal solution. Instead, generic programming introduces dedicated *interface objects* that explicitly decouple algorithms and data structures from each other. For our present purpose, the most important interface objects are *iterators* [5]. Iterators encapsulate how to navigate on a container data structure, that is, they point to a specific element (pixels) within a specific container and can be moved to refer to others.

In the Standard Template Library (STL), Musser and Stepanov define five iterator categories that completely cover the needs of algorithms

[2]In practice, many compilers will generate slightly slower code for generic algorithms because current optimizers are more familiar with pointers than with genericity. There is, however, no fundamental reason why generic algorithms must be slower, and some compilers (such as KAI C++) can already generate equally fast code in both cases.

operating on top of linear (1-D) data structures. These iterator categories have been developed on the basis of a thorough analysis of what different algorithms need to do their tasks, and what different data structures can efficiently implement. For our present discussion, three categories are of importance:

Forward iterators allow going forward to the next element (++iterator), to find out if two iterators point to the same position (`iterator1 == iterator2`) and to access the data at the current position (`*iterator`). This functionality is sufficient to implement the `copy()` algorithm.

Bidirectional iterators also provide a function to go backwards to the previous element (−−iterator). This functionality would be needed to implement a `reverseCopy()` function.

Random-access iterators additionally allow jumping back and forth by arbitrary offsets (`iterator += n, iterator -= n`), to compare iterators according to an ordering (`iterator1 < iterator2`), and to access an element at a given distance of the current position (`iterator[n]`). This functionality is required by most sorting algorithms, for example, `quicksort()`.

Typically, a singly linked list provides forward, and a doubly linked list bidirectional iterators, while a vector supports random-access iteration. It can be seen that the iterators adopt the syntax of C pointers, which always conform to the requirements of random-access iterators. Using iterators, the copy function for linear (1-D) data structures will look like this:

```
template <typename SrcIterator, typename DestIterator>
void copy(SrcIterator s, SrcIterator end, DestIterator j)
{
    for(; s != end; ++s, ++d)   *d = *s;
}
```

By convention, the iterator `end` marks the end of the iteration by pointing to the first position *past the sequence*. In Section 6.3, we will generalize these iterator categories to two-dimensions, that is, images.

6.2.4 Functors

To improve the flexibility of algorithms, Stepanov and Musser have introduced *functors*. Functors encapsulate an important subcomputation, which we want to vary independently of the enclosing algorithm. We can exchange the respective subcomputations by simply passing different functors to the algorithm. For illustration, consider the `transform()` algorithm. We can interpret `transform()` as a generalization of `copy()`: while `copy()` writes the source data into the destination sequence unchanged, `transform()` performs an arbitrary transformation in between. As there are infinitely many possible transformations we encapsulate them into a functor so that a single `transform()` template can be used for all possible transformations:

```
template <typename SrcIterator, typename DestIterator, typename Functor>
void transform(SrcIterator s, SrcIterator end
               DestIterator d, Functor func)
{                                  // ^^^^ pass functor
    for(; s != end; ++s, ++d)   *d = func(*s);
}                                // ^^^^ apply functor
```

A functor behaves like a function pointer, but it is both more powerful (because it can store internal state) and more efficient (because the compiler can inline it). We will see more examples for functors when we implement the basic image processing functionality in Section 6.5.2.

6.2.5 Data accessors

The idioms reported so far (generic data structures and algorithms, iterators, and functors) are the basis of the STL. However, experience has shown that algorithms and data structures are not sufficiently independent as long as we have only a single function

`iterator::operator*()`

to access the iterator's current data item. There are two fundamental problems with this function: First, it assumes that all algorithms want to see the entire data at the current position. This makes it problematic to operate on a subset of the actual data structure (such as a single color band of an RGB image). Second, `*iterator` must be used for both reading and writing the data. This means, `operator*()` must return the data by reference if we want to overwrite it. This is difficult to implement for certain container implementations (e. g., a multiband RGB image).

Therefore, additional interface objects, called *data accessors*, are needed to encapsulate actual data access [6]. A data accessor reads data at the current iterator position via a `get()` function, and writes them via a `set()` function. Within these functions, arbitrary adaptations (such as extracting just the red band of an RGB pixel, or transforming RGB to gray) can be performed. In general, a data accessor looks like this:

```
template <typename VALUETYPE, typename ITERATOR>
struct SomeAccessor
{
    typedef VALUETYPE value_type;  // advertise your value type

    value_type get(ITERATOR & i) const;   // read an item
    void set(value_type v, ITERATOR & i) const;  // write an item
};
```

Using this data accessor, we can implement a more general version of the linear `copy()` algorithm like this:

```
template <typename SrcIterator, typename SrcAccessor,
          typename DestIterator, typename SrcAccessor>
void copy(SrcIterator s, SrcIterator end, SrcAccessor sa,
          DestIterator d, DestAccessor da)
{
    for(; s != end; ++s, ++d)
        da.set(sa.get(s), d); // read/write via accessor
}
```

Although it is probably hard to see the advantages of accessors (and generic programming in general) in an algorithm as simple as `copy()`, the same techniques can be applied to arbitrary complicated algorithms, where reuse will really pay off.

6.2.6 Meta-information: traits

To automate instantiation of more complicated generic algorithms, we also need meta-information, that is, machine readable descriptions of the properties and constraints of generic building blocks. In C++, meta-information is commonly stored in so-called *traits* classes [7]. We illustrate traits using two important examples.

As is well known, C++ performs certain automatic *type conversions*, if an expression can not be evaluated otherwise. For example, integers are promoted to double if they are mixed with real numbers in an arithmetic expression. In case of generic algorithms, we can not know in advance which type of conversions should be applied. Traits can be used to provide the meta-information needed in such a situation. We define `PromoteTraits`, which describe the type promotion rules between two types `T1` and `T2` as follows:

```
template <typename T1, typename T2> struct PromoteTraits;

struct PromoteTraits<float, unsigned char> {
    typedef float Promote;   // unsigned char is promoted to float
};

struct PromoteTraits<RGBStruct<float>, RGBStruct<unsigned char> > {
   typedef RGBStruct<float> Promote; // RGBStruct<unsigned char> is
};                                   // promoted to RGBStruct<float>
```

These traits classes must be defined for all type combinations we are interested in. Then we can use RGBStruct-traits to implement a generic function for, say, component-wise addition of two `RGBStructs`:

```
template <typename T1, typename T2>
PromoteTraits<RGBStruct<T1>, RGBStruct<T2> >::Promote
operator+(RGBStruct<T1> const & t1, RGBStruct<T2> const & t2)
{
    // construct result from promoted sums of color components
    return PromoteTraits<RGBStruct<T1>, RGBStruct<T2> >::Promote(
          t1.red + t2.red, t1.green + t2.green, t1.blue + t2.blue);
}
```

The traits class determines the type of the result of the addition for any combination of `RGBStruct<T1>` and `RGBStruct<T2>` for which we have defined `PromoteTraits`. In addition to binary promotion traits, we also define unary traits that are used to determine the type of temporary variables for intermediate results (including the initialization of these variables). These definitions look like this:

```
template <typename T> struct NumericTraits;

struct NumericTraits<unsigned char> {
    // store intermediate results as float
    typedef float Promote;

    // neutral element of addition
    static unsigned char zero() { return 0; }
    // neutral element of multiplication
    static unsigned char one() { return 1; }
};

struct NumericTraits<RGBStruct<unsigned char> > {
    typedef RGBStruct<float> Promote;

    static RGBStruct<unsigned char> zero() {
        return RGBStruct<unsigned char>(0, 0, 0);
    }
    static RGBStruct<unsigned char> one() {
        return RGBStruct<unsigned char>(1, 1, 1);
    }
};
```

The necessity of using `NumericTraits` is best seen in a function calculating the average of some values. Look at the following traditional code:

```
unsigned char value[20] = {...};

float average = 0.0;
for(int i=0; i<20, ++i)
            average += value[i];
average /= 20.0;
```

Had we stored the average in the original data type `unsigned char`, a numeric overflow would probably have occurred after a few additions, leading to a wrong result. We will use `NumericTraits` to avoid this problem. The generic implementation of the averaging algorithm on a 1-D sequence will look like this:

```
template <typename Iterator, typename Accessor>
NumericTraits<Accessor::value_type>::Promote
average(Iterator i, Iterator end, Accessor a)
{
    NumericTraits<Accessor::value_type>::Promote sum =
                  NumericTraits<Accessor::value_type>::zero();

    int count = 0;
    for(; i != end; ++i, ++count)  sum = sum + a.get(i);

    return (1.0 / count)  * sum;
}
```

6.3 Two-dimensional iterators

6.3.1 Navigation

Although linear iterators as described in the previous section have their place in computer vision (see Section 6.7), we will first and foremost work on 2-D images. Thus, we must generalize the iterator concept to two dimensions (higher dimensions can be introduced analogously, but will not be covered in this chapter), that is, we must define navigation functions that tell the iterator in which coordinate direction to move. As a first idea, we could define navigation functions like `incX()` or `subtractY()`, but in C++ there exists a more elegant solution that uses nested classes to define different views onto the same navigation data. More specifically, consider the following iterator design:

```
class ImageIterator {
  public:
    // ...

    class MoveX {
        // data necessary to navigate in X direction
      public:
        // navigation function applies to X-coordinate
        void operator++();
        // ...
    };

    class MoveY {
        // data necessary to navigate in Y direction
      public:
        // navigation function applies to Y-coordinate
        void operator++();
        // ...
    };

    MoveX x;  // x-view to navigation data
    MoveY y;  // y-view to navigation data
};
```

Now we can use the nested objects `x` and `y` to specify the direction to move along as if we had two 1-D iterators:

```
ImageIterator i;
++i.x;  // move in x direction
++i.y;  // move in y direction
```

This way we can define 2-D iterators in terms of the same operations which the STL defines for 1-D iterators. Table 6.1 lists the complete functionality to be supported by 2-D random access iterators[3]. A standard implementation that works with all image formats that internally store image data as a linear memory block is contained on the CD-ROM that accompanies this book.

[3]Forward and bidirectional image iterators can easily be defined by dropping some of the listed functions.

Table 6.1: *Requirements for image iterators (Meaning of symbols: ImageIterator i, j; int dx, dy; struct Dist2D { int width, height; } dist;)*

Operation	Result	Semantics
ImageIterator::MoveX		type of the x navigator
ImageIterator::MoveY		type of the y navigator
ImageIterator::value_type		type of the pixels
++i.x; i.x++	void	increment x coordinate[1,3]
--i.x; i.x--	void	decrement x coordinate[1,3]
i.x += dx	ImageIterator::MoveX&	add dx to x coordinate
i.x -= dx	ImageIterator::MoveX&	subtract dx from x coord.
i.x - j.x	int	difference of x coordinates[2]
i.x = j.x	ImageIterator::MoveX&	i.x += j.x - i.x[2]
i.x == j.x	bool	j.x-i.x == 0[1,2]
i.x < j.x	bool	j.x-i.x > 0[2]
++i.y; i.y++	void	increment y coordinate[3]
--i.y; i.y--	void	decrement y coordinate[3]
i.y += dy	ImageIterator::MoveY&	add dy to y coordinate
i.y -= dy	ImageIterator::MoveY&	subtract dy from y coord.
i.y - j.y	int	difference of y coordinates[2]
i.y = j.y	ImageIterator::MoveY&	i.y += j.y - i.y[2]
i.y == j.y	bool	j.y-i.y == 0[2]
i.y < j.y	bool	j.y-i.y > 0[2]
ImageIterator k(i)		copy constructor
k = i	ImageIterator&	copy assignment
i += dist	ImageIterator&	add offset to x and y
i -= dist	ImageIterator&	subtract offset from x and y
i + dist	ImageIterator	add offset to x and y
i - dist	ImageIterator	subtract offset from x and y
i - j	Dist2D	difference in x and y[2]
i == j	bool	i.x==j.x && i.y==j.y[2]
*i	value_type &	access the pixel data[4]
*i	value_type	read the pixel data[5]
i(dx, dy)	value_type &	access data at offset (dx, dy)[4,6]
i(dx, dy)	value_type	read data at offset (dx, dy)[5,6]

[1]prefer this in inner loops
[2]i and j must refer to the same image
[3]*i.x++, *i.y++ etc. not allowed
[4]mutable iterator only, optional
[5]constant iterator only, optional
[6]replaces operator[] in 2-D

All functions listed must execute in constant time. To further optimize execution speed, we also label some functions as “prefer this in inner loops,” which means that these functions should be fastest (i. e., have the smallest “big O factor”). Typically, they should consist of just one addition respective comparison to match the speed of normal pointer arithmetic. All navigation functions must commute with each other, that is, the same location must be reached if a sequence of navigation functions (say `++i.x; ++i.y;`) is applied in a different order (like `++i.y; ++i.x;`).

6.3.2 Direct data access

In contrast to the STL, the image iterator functions for direct data access (`operator*` and `operator()`) have been made *optional*, because there is no guarantee that these functions can be defined as desired when we want to implement iterators for *existing data structures* that we can not change. For example, suppose for some reason we must use the `AbstractImage` defined in the introduction. To make it compatible with our generic algorithms, we need to wrap it into an image iterator. This is actually very easy, because we can use plain integers for the iterators’ `MoveX` and `MoveY` members:

```
struct AbstractImageIterator
{
    typedef unsigned char value_type;
    typedef int           MoveX;
    typedef int           MoveY;
    int x, y;

    // init iterator at the upper left corner of the image
    AbstractImageIterator(AbstractImage * img)
    : image_(img), x(0), y(0)
    {}

    // point two iterators to different positions ?
    bool operator!=(AbstractImageIterator const & i) const
    {
        return (x != i.x || y != i.y);
    }
    // ... not all functions shown

    // delegate data access to the underlying image
    unsigned char get() const
    {
        return image_->getPixel(x, y);
    }

    void set(unsigned char v)
    {
        image_->setPixel(v, x, y);
    }

    /* these functions are hard or impossible to implement:
    unsigned char & operator*();
    unsigned char & operator()(int dx, int dy);
    */
  private:
    AbstractImage * image_;
};
```

For a mutable iterator, `operator*` and `operator()` are required to return a reference to the current data item. However, as the underlying `AbstractImage` does not return data by reference, the iterator can not do it either (Tricks to work around this problem, such as the proxy technique described in Meyers [8], provide partial solutions at best). Therefore, in Table 6.1 these functions are marked "optional." A general and satisfying solution to this problem is a major motivation behind the introduction of *data accessors* in Section 6.4.

6.3.3 Image boundaries

In contrast to a 1-D sequence, which has just 2 ends, an image of size $w \times h$ has $2(w + h + 2)$ past-the-border pixels[4]. To completely mark the boundaries of this image, we need a set of iterators — one for the begin and end of each row and each column. Iterators that refer to past-the-border pixels will be called *boundary markers* (no new type is required). According to their position we distinguish *left*, *right*, *top*, and *bottom* boundary markers. It is not desirable to pass a set of $2(w + h + 2)$ boundary markers to an algorithm. Therefore we specify rules as to how the algorithm can create any boundary marker from just a few on known positions.

All 2-D iterators must be able to navigate on past-the-border locations as if these pixels were inside the image. Thus we can transform an iterator into a boundary marker and a boundary marker into another one. The difference is that boundary markers must not be lost, that is, we are not allowed to access data at past-the-border positions. To be exact, top boundary markers are defined as follows:

- For each column a unique top boundary marker exists. A program may hold many identical instances of this marker.
- Applying `++i.y` will move the boundary marker to the first pixel of the current column, a subsequent `--i.y` recreates the marker. Likewise, `i.y+=dy` may be applied if the target pixel is inside the image or past the bottom border ($dy \leq h+1$). Subsequent `i.y-=dy` recreates the marker.
- Applying `++i.x`, `--i.x`, `i.x+=dx` or `i.x-=dx` produces the top boundary markers of the respective target columns, if they exist.

Analogous definitions apply to the other boundaries. Examples for generic algorithms implemented on the basis of these definitions are given in what follows.

[4]Past-the-border pixels are the outside pixels having at least one 8-neighbor in the image.

6.4 Image data accessors

When we implemented an image iterator for the `AbstractImage` we saw that it was impossible to provide data access operators (`operator*` and `operator()`) that returned the current data item by reference. Another common example for a data structure suffering from this problem is the *multiband RGB image*. In a multiband RGB image, pixel values are stored in three separate images (bands) for each of the three color components, rather than in one image of compound RGBStructs (as, for example, the `RGBStructImage` defined in section “Generic programming”). A typical implementation of a *multiband RGB image* could look like this:

```
class MultibandRGBImage
{
  public:
    unsigned char & red(int x, int y);
    unsigned char & green(int x, int y);
    unsigned char & blue(int x, int y);
    //...
  private:
    unsigned char * redband, * greenband, * blueband;
};
```

The corresponding image iterator would simply mirror the access functions of the image:

```
struct MultibandRGBImageIterator
{
    // navigation functions not shown ...

    unsigned char & red();
    unsigned char & green();
    unsigned char & blue();

    /* this is difficult or impossible to implement
    RGBStruct<unsigned char> & operator*();
    RGBStruct<unsigned char> & operator()(int dx, int dy);
    */
};
```

Once again we can not implement the standard access functions used within the STL, because the desired return type

`RGBStruct<unsigned char>&`

does not physically exist in the underlying image data structure. Thus, while it is easy to define a uniform navigation interface for the iterators, we can not guarantee that all iterators have a uniform data access interface. Kühl and Weihe [6] proposed an additional level of indirection, the *data accessor*, to recover the desired uniform interface. As was mentioned earlier, data accessors provide a pair of `get()` and `set()` functions that are inlined by the compiler so that the additional indirection does not affect performance. In cases where the iterator provides `operator*`, `get` and `set` simply call it:

```
template <typename VALUETYPE, typename STANDARDITERATOR>
struct StandardAccessor
{
    typedef VALUETYPE value_type;  // advertise your value type

    value_type get(STANDARDITERATOR & i) const {
        return *i;   // read current item
    }

    void set(value_type v, STANDARDITERATOR & i) const {
        *i = v;      // write current item
};
```

Because the `AbstractImageIterator` does not work this way, it needs a different accessor that could look like this:

```
struct AbstractImageAccessor
{
    typedef unsigned char value_type;

    value_type get(AbstractImageIterator & i) const {
        return i.get();
    }
    void set(value_type v, AbstractImageIterator & i) const {
        i.set(v);
    }
};
```

In addition to the standard `get()` and `set()` functions, an RGB accessor provides functions to access each color separately. These functions will be used by algorithms that explicitly require RGB pixels. For the multiband RGB image, such an accessor could be defined as follows:

```
struct MultibandRGBImageAccessor
{
    typedef RGBStruct<unsigned char> value_type;
    typedef unsigned char component_type;

    value_type get(MultibandRGBImageIterator & i) const {
        return value_type(i.red(), i.green(), i.blue());
    }

    void set(value_type rgb, MultibandRGBImageIterator & i) const {
        i.red() = v.red;      // assume that the iterator
        i.reen() = v.green;   // mirrors the red(), green(),
        i.blue() = v.blue;    // blue() function of the image
    }                          // which return data by reference

    component_type getRed(MultibandRGBImageIterator & i) const {
        return i.red();
    }

    void setRed(component_type v, MultibandRGBImageIterator & i) const {
        i.red() = v;
    }
    // ...
};
```

Providing a uniform data access interface regardless of the underlying image implementation is not the only advantage of introducing accessors: they can also be used to perform an arbitrary transformation before the algorithm actually sees the data (for instance, color to gray conversion), or to restrict access to just a part of the current item, such as one color component of an RGB pixel. For example, we could select just the red band of the `RGBStructImage` by using the following accessor:

```
struct RedComponentAccessor
{
    typedef unsigned char value_type;

    value_type get(RGBStructImage::Iterator & i) {
        return (*i).red;
    }

    void set(value_type v, RGBStructImage::Iterator & i) const {
        (*i).red = v;
    }
};
```

For example, we could now apply a scalar convolution to each color component separately by calling the convolution algorithm (see Section 6.5.3) with red, green, and blue band accessors in turn. Without accessors, this behavior would be difficult to implement for the `RGBStructImage`, whereas it would be easy for a `MultibandRGBImage`. Hence accessors serve to smooth out the differences between various implementation choices for the image data structures, which would otherwise make universally reusable algorithm implementations impossible.

6.5 Generic algorithms on images

6.5.1 Basic image processing algorithms

Now that we have covered all preliminaries we can finally come to the real thing, the implementation of generic algorithms on images. Basic image processing algorithms will be direct generalizations of the 1-D algorithms described in Section “Generic Programming.” The region of interest will be determined by a pair of iterators, called `upperleft` and `lowerright`. Similarly to the end iterator in 1-D, which points to the first position *past* the desired range, `lowerright` denotes the first pixel *outside* the desired ROI (i. e., one pixel right and below of the last pixel inside the ROI), while `upperleft` points to the first pixel in the ROI (see Fig. 6.1). Note, that `upperleft` and `lowerright` may refer to arbitrary positions in the image, so that any algorithm can be applied to any rectangular subregion of an image without modification.

The basic image processing functionality can be implemented with just five generic functions:

initImage()	inspectImage()	inspectTwoImages()
copyImage()	transformImage()	combineTwoImages()

With the exception of `copyImage()`, all these functions take functors to define their actual semantics (see next section). The first is used to load

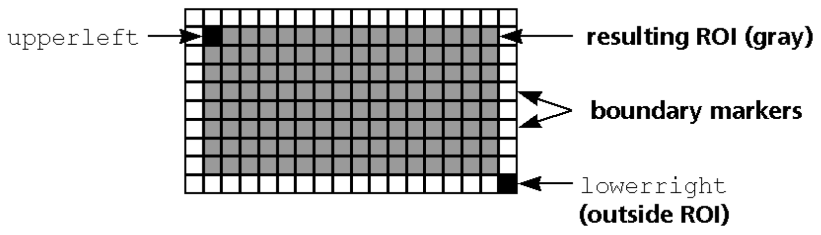


Figure 6.1: *Iterators at* `upperleft` *and* `lowerright` *determine the region of interest.*

specific, possibly location dependent, values in a new image. The next two are applied to collect statistics for the entire image resp. labeled regions. As the name suggests, `copyImage()` is used to copy images (which may require an automatic type conversion). As we can pass iterators at arbitrary positions, `copyImage()` also covers the functionality of extracting and inserting subimages. Finally, the last two functions will be used to perform point operations involving one image (such as thresholding or histogram manipulation) and two images (such as pixel arithmetic), respectively. To give an idea of how these functions work, we will give the implementation of two of them:

```
template <class SrcIterator, class SrcAccessor, class Functor>
void inspectImage(SrcIterator upperleft, SrcIterator lowerright,
                  SrcAccessor sa, Functor & func)
{
    // iterate down first column
    for(; upperleft.y < lowerright.y; ++upperleft.y)
    {
        SrcIterator s(upperleft);   // s points to begin of row
        for(; s.x < lowerright.x; ++s.x)   // across current row
                           func(sa.get(s));    // apply functor
    }
}

template <class SrcIterator, class SrcAccessor,
          class DestIterator, class DestAccessor, class Functor>
void transformImage(
    SrcIterator supperleft, SrcIterator lowerright, SrcAccessor sa,
    DestIterator dupperleft, DestAccessor da, Functor func)
{
    for(; supperleft.y < lowerright.y;      // down first column
                          ++supperleft.y, ++dupperleft.y)
    {
        SrcIterator s(supperleft);  // s and d point to begin of
        DestIterator d(dupperleft); // current src resp. dest row
        for(; s.x < lowerright.x; ++s.x, ++d.x)     // accross row
        {
            da.set(func(sa.get(s)), d);         // apply functor
        }
    }
}
```

To extend the functionality towards arbitrary *regions of interest* (ROIs) we also introduce versions of these algorithms that use mask images. Application of these algorithms' functionality is restricted to the positions where the mask returns true. These algorithms are denoted by the suffix `'If'`. `copyImageIf()` looks like this:

```
template <class SrcIterator, class SrcAccessor,
          class MaskIterator, class MaskAccessor,
          class DestIterator, class DestAccessor>
void copyImageIf(
    SrcIterator supperleft, SrcIterator lowerright, SrcAccessor sa,
    MaskIterator mupperleft, MaskAccessor ma,
    DestIterator dupperleft, DestAccessor da)
{
    for(; supperleft.y < lowerright.y;
             ++supperleft.y, ++dupperleft.y, ++mupperleft.y)
    {
        SrcIterator s(supperleft);
        MaskIterator m(mupperleft);
        DestIterator d(dupperleft);
        for(; s.x < loweright.x; ++s.x, ++d.x, ++m.x)
        {
            if(ma.get(m))              // check the mask
                da.set(sa.get(s), d);  // copy only if true
        }
    }
}
```

In conjunction with various functors (see next section), these functions are extremely versatile. By passing different iterators and accessors, we can apply them to

- arbitrary pixel data types (`byte`, `int`, `float`, `RGBStruct` or anything else),
- arbitrary image data structures (`GenericImage`, `AbstractImage`, `MultibandRGBImage` or anything else),
- subsets of structured pixel types (For example, by passing a `RedBandAccessor` and a `GreenBandAccessor` to `copyImage()`, we can copy the red band of an RGB image into the green band of another.),
- debug versions of the data structures (For example, we can use a bounds-checking iterator during debugging.), and
- arbitrary subimages (by passing iterators at suitable positions and/or use mask images).

Neither of these options requires any changes to the algorithms. Thus, we need much less source code than a traditional library to provide the same functionality. In our generic library VIGRA (which is provided in `/software/04` on the CD-ROM accompanying this book), the algorithms described here, plus functors for all common pixel transformations (such as thresholding, algebraic functions, pixel arithmetic, and logical functions), take less than 10 Kbytes of source code. In contrast, in a traditional system like KHOROS, all these functions must be implemented repeatedly, which consumes over 100 Kbytes of source code. These numbers directly translate into a huge gain in programmer productivity.

6.5.2 Functors for image processing

To bring the forementioned functions to life, we need to implement functors for common operations. For example, consider an algorithm that maps an image from an arbitrary scalar intensity domain into the displayable range 0...255. To do this we first need to find the minimal and maximal pixel values in the original image. This can be done by the function `inspectImage()` and the following `FindMinMaxFunctor`[5]:

```
template <class PIXELTYPE>
struct FindMinMaxFunctor
{
    MinmaxFunctor() : count(0) {}

    void operator()(PIXELTYPE const & i)
    {
        if(count == 0) {
            max = min = i;  // first pixel: init max and min
        }
        else {
            if(max < i) max = i;   // update max
            if(i < min) min = i;   // and min
        }
        ++count;   // update pixel count
    }

    int count;
    PIXELTYPE min, max;
};

Image img(width, height);
//...
FindMinMaxFunctor<Image::PixelType>minmax;

inspectImage(img.upperLeft(), img.lowerRight(), img.accessor(),
             minmax);
```

Now that we know the minimum and maximum, we can apply a linear transform to map the original domain onto the range 0...255. We use the function `transformImage()` in conjunction with a functor `LinearIntensityTransform`:

```
template <class PIXELTYPE>
struct LinearIntensityTransform
{
    typedef typename NumericTraits<PIXELTYPE>::Promote value_type;

    LinearIntensityTransform(value_type scale, value_type offset)
    : scale_(scale), offset_(offset)
    {}

    value_type operator()(PIXELTYPE const & i) const
    {
        return scale_ * (i + offset_);
    }
```

[5]For convenience, in the sequel we will assume that we are using an image data type that has member functions `upperLeft()`, `lowerRight()`, and `accessor()`, as well as embedded `typedefs PixelType, Iterator` and `Accessor`. Also, the iterator shall support `operator*`.

```
    value_type scale_, offset_;
};

//...

LinearIntensityTransform<Image::PixelType>
         map_intensity(255.0*(minmax.max-minmax.min), -minmax.min);

Image destimg(width, height);

transformImage(img.upperLeft(), img.lowerRight(), img.accessor(),
           destimg.upperLeft(), destimg.accessor(), map_intensity);
```

At first this looks rather complicated and lengthy. One should, however, keep in mind that the functions and functors need to be implemented only once, and can then be reused for any image we want to transform, regardless of the pixel types and implementation details. Moreover, some further simplifications to the foregoing code are possible. For example, we can define factory functions to generate the triples `[upperleft, loweright, accessor]` respectively pairs `[upperleft, accessor]` automatically, given an image. By using expression templates, we can also let the compiler generate functors automatically, given the expression it implements. However, a description of these simplifications is beyond the scope of this chapter.

6.5.3 Higher-level functions

The techniques outlined here are likewise applied to implement higher-level algorithms. Currently, we have implemented about 50 generic algorithms in our library VIGRA. This includes edge and corner detection, region growing, connected components labeling, feature characterization, and calculation of feature adjacency graphs. A few examples shall illustrate some possibilities; a larger selection of examples can be found on the CD-ROM provided with this book.

First let us look at a simple averaging algorithm. We will give its full implementation to illustrate typical image iterator usage in operations involving a window around the current location. This implementation was also used as the generic variant in the benchmark tests reported in the next section[6].

```
template <class SrcImageIterator, class SrcAccessor,
          class DestImageIterator, class DestAccessor>
void averagingFilter(SrcImageIterator supperleft,
                     SrcImageIterator slowerright, SrcAccessor sa,
                     DestImageIterator dupperleft, DestAccessor da,
                     int window_radius)
{
    // determine pixel type of source image
    typedef typename SrcAccessor::value_type SrcType;

    // determine temporary type for averaging  (use
```

[6]Of course, in practice one would use a separable algorithm to gain speed. Section 6.7.1 shows how this is effectively done using row and column iterator adapters.

```
        // NumericTraits::Promote to prevent overflow)
        typedef typename NumericTraits<SrcType>::Promote SumType;

        int width = slowerright.x - supperleft.x;  // calculate size
        int height = slowerright.y - supperleft.y; // of image

        for(int y=0; y < height;  // down first column
                        ++y, ++supperleft.y, ++dupperleft.y)
        {
            SrcImageIterator xs(supperleft);   // create iterators
            DestImageIterator xd(dupperleft);  // for current row

            for(int x=0; x < width; ++x, ++xs.x, ++xd.x)
            {
                // clip the window
                int x0 = min(x, window_radius);
                int y0 = min(y, window_radius);
                int winwidth  = min(width - x, window_radius + 1) + x0;
                int winheight = min(height - y, window_radius +1) + y0;

                // init sum to zero
                SumType sum = NumericTraiTs<SrcType>::zero();

                // create y iterators for the clipped window
                SrcImageIterator yw(xs - Dist2D(x0, y0)),
                                 ywend(yw + Dist2D(0, winheight));

                for(; yw.y != ywend.y; ++yw.y)
                {
                    // create x iterators for the clipped window
                    SrcImageIterator xw(yw),
                                     xwend(xw + Dist2D(winwidth, 0));

                    // fastest operations in inner loop
                    for(; xw.x != xwend.x; ++xw.x)
                    {
                        sum += sa.get(xw);
                    }
                }

                // store average in destination
                da.set(sum / (winwidth * winheight), xd);
            }
        }
    }
```

The algorithm calculates the average of all pixel values in a square window (whose size is given by `window_radius` in chess-board metric). If the window is partially outside the image, it is clipped accordingly. The result is written into the destination image. Note that `NumericTraits` are used to determine the type of the variable to hold the sum of the pixel values to prevent overflow (which would almost certainly occur if `SrcAccessor::value_type` were `unsigned char`).

This algorithm is easily generalized to arbitrary convolutions. Similarly to images, we pass convolution kernels by iterators and accessors. This allows us to represent kernels in many different ways—we can use dedicated kernel objects as well as images and even C-style arrays, without requiring any change to the algorithm. In contrast to images, we must also pass the kernels' center. Therefore, the declaration of a general convolution function could look like this:

```
template <class SrcIterator, class SrcAccessor,
          class DestIterator, class DestAccessor,
          class KernelIterator, class KernelAccessor>
void convolveImage(SrcIterator sul, SrcIterator slr, SrcAccessor sa,
                   DestIterator dul, DestAccessor da,
                   KernelIterator kupperleft, KernelIterator kcenter,
                   KernelIterator klowerright, KernelAccessor ka);
```

Seeded region growing is a particularly good example for the versatility of the generic approach. In general, seeded region growing is defined by a set of seed regions, an image containing pixel information (like gray levels, colors, gradients etc.), and a statistics functor to determine into which region each pixel fits best. Thus, the corresponding function declaration would look as follows:

```
template <class SrcIterator, class SrcAccessor,
          class SeedIterator, class SeedAccessor,
          class DestIterator, class DestAccessor,
          class StatisticsFunctor>
void seededRegionGrowing(SrcIterator sul, SrcIterator slr, SrcAccessor sa,
                         SeedIterator seedul, SeedAccessor seeda,
                         DestIterator dul, DestAccessor da,
                         StatisticsFunctor statfunc);
```

Without any modification of this algorithm's source code, we can use it to implement many different region growing variants by simply passing different images and functors. Possibilities include:

- If the statistics functor returns always the same value, a morphological dilation of all regions will be performed until the image plane is completely covered. In particular, if the seed regions are single pixels, the Voronoi tessellation will be determined.
- If the source image contains gradient magnitudes, the seed image contains gradient minima, and the statistics functor returns the local gradient at every location, the watershed algorithm will be realized.
- If the statistics functor calculates and constantly updates region statistics for every region (such as mean gray values or colors), it can compare every pixel with its neighboring regions to determine where it fits best.

This illustrates a general experience we have made with genericity: Due to the versatility of generic functions, we do not need special implementations for every variation of a general algorithm. This means that a smaller number of generic functions is sufficient to implement functionality analogous to a traditional image analysis library. Also, generic functions are very robust under algorithm evolution: if better growing criteria are found, we simply pass other functors and/or images, but the core implementation remains unchanged.

6.6 Performance

To assess the claims about the performance of generic algorithms we conducted a number of benchmark tests. We tested four different implementations of the averaging algorithm for a 2000×1000 float image with window size 7×7 on various systems. These are the implementation variants we tested:

1. a hand-optimized pointer-based version;
2. a traditional object-oriented variant using the abstract image interface with a virtual access function as defined in `AbstractImage` in the introduction;
3. the averaging algorithm as shown in the previous section using the `AbstractImageIterator` which wraps the virtual function of `AbstractImage`; and
4. the averaging algorithm as shown in the previous section using the standard implementation for image iterators operating on linear memory (see `/software/06` on the CD-ROM).

The results of these tests are given in Table 6.2. The table shows that with the best compilers the proposed image iterator (variant 4) comes close to the performance of a hand-crafted algorithm (variant 1), whereas the versions using the abstract image interface (virtual function calls) are much slower (variants 2 and 3). Even in the worst case `variant 4` takes less than double the optimal time. In the light of the flexibility gained this should be acceptable for many applications. Moreover, there is no principal reason that our iterator implementation must be slower than the hand-crafted version. Thus, with the permanent improvement of the compiler optimizers, we expect the performance of our iterator to increase further.

6.7 Image iterator adapters

On the basis of the proposed iterator interface we can define iterators that modify the standard behavior in various ways. In many cases this allows for improving reusability by turning *algorithms* into *iterator adapters*. For example, algorithms that need 1-D projections of the image (i.e., a particular order in which all or some pixels are visited) can be generalized by encapsulating the particular projection into appropriate iterator adapters. By exchanging iterator adapters, different access sequences can be realized without changing the actual algorithm.

The simplest 1-D iterator adapter is, of course, the scan line iterator, which scans the entire image in scan line order. It can often be implemented as a normal pointer to the raw image data and this makes

Table 6.2: *Performance measurements for different implementations of the averaging algorithm*

	Implementation Variant			
System	1	2	3	4
1	2.75 s (100%)	12.6 s (460%)	7.5 s (270%)	3.3 s (120%)
2	9.9 s (100%)	42.2 s (425%)	35.6 s (360%)	18.5 s (190%)
3	11.5 s (100%)	64.8 s (560%)	55.7 s (480%)	17.0 s (150%)
4	8.1 s (100%)	38.4 s (470%)	13.5 s (170%)	9.1 s (112%)

System 1	SGI O2, IRIX 6.3, SGI C++ 7.1
System 2	SGI INDY, IRIX 5.3, SGI C++ 4.0 (outdated compiler)
System 3	Sun SparcServer 1000, Solaris 5.5, Sun C++ 4.2
System 4	PC Pentium 90, Windows NT 4.0, Microsoft VC++ 4.0

it very fast. Thus it is best suited for internal copy and initialization operations, when we need to process the entire image very fast.

6.7.1 Row and column iterators

Row and column iterators are very useful iterator adapters because they support the implementation of separable algorithms. They provide an interface compatible with the STL's random access iterators and restrict navigation to the specified row or column:

```
template <class ImageIterator>
struct RowIterator
{
    RowIterator(ImageIterator const & i)  // position of iterator i
    : iter(i)                             // determines row to operate on
    {}

    RowIterator & operator++() {
        iter.x++;     // go always in x direction
        return *this;
    }

    value_type & operator*() {
        return *iter; // delegate access to enclosed image iterator
    }
    // etc...

    ImageIterator iter;
};
```

Now, when we want to define a *separable algorithm*, just one implementation is sufficient for both directions (and in fact for any 1-D projection), because the actual direction is specified solely by the type of the iterator adapter passed to the algorithm. Consider the following 1-D algorithm for a recursive exponential filter:

```
template <class Src1DIterator, class SrcAccessor,
          class Dest1DIterator, class DestAccessor>
void recursiveSmoothSignal(
           Src1DIterator sbegin, Src1DIterator end, SrcAccessor sa,
           Dest1DIterator dbegin, DestAccessor da, float scale)
{
   ...
}
```

If we recall that this algorithm can operate in-place (i. e., source and destination signal can be identical, so that no temporary signal must be generated), we can implement the 2-D recursive filter by simply calling the 1-D version twice, first using row iterators, then using column iterators:

```
template <class SrcImageIterator, class SrcAccessor,
          class DestImageIterator, class DestAccessor>
void recursiveSmoothImage(
        SrcImageIterator sul, SrcImageIterator slr, SrcAccessor sa,
        DestImageterator dul, DestAccessor da, float scale)
{
    int width = slr.x - sul.x;  // calculate size
    int height = slr.y - sul.y; // of image

    DestImageIterator destul = dul;  // save iterator for later use

    for(int y=0, y<height, ++sul.y, ++dul.y, ++y)
    {
        RowIterator<SrcImageIterator> sr(sul);  // create iterator
        RowIterator<DestImageIterator> dr(dul); // adapters for row

        // convolve current row
        recursiveSmoothSignal(sr, sr+width, sa, dr, da, scale);
    }

    for(int x=0, dul=destul; x<width; ++dul.x, ++x)
    {
        ColumnIterator<DestImageIterator> dc(dul); // adapter for column

        // convolve current column (in-place)
        recursiveSmoothSignal(dc, dc+height, da, dc, da, scale);
    }
}
```

6.7.2 Iterator adapters for arbitrary shapes

The idea of row and column iterators is easily generalized to arbitrarily shaped linear structures such as lines, rectangles, ellipses, and splines. For example, we can turn Bresenham's algorithm for drawing a straight line into an iterator adapter like this:

```
template <class ImageIterator>
struct BresenhamLineIterator
{
    BresenhamLineIterator(ImageIterator start, ImageIterator end)
    : iterator_(start), x_(0.0), y_(0.0)
    {
       int dx = end.x - start.x;  // calculate the distance
       int dy = end.y - start.y;  // from start to end
       int dd = max(abs(dx), abs(dy)) > 0 ? max(abs(dx), abs(dy) : 1;
       dx_ = (float)dx / abs(dd);  // init the appropriate
```

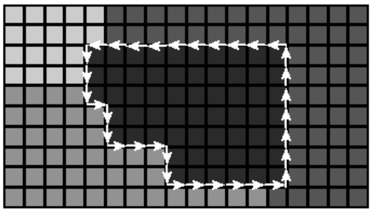


Figure 6.2: A ContourIterator (white arrows) follows the contour of the darkest region.

```
        dy_ = (float)dy / abs(dd);  // increments (Bresenham algorithm)
    }

    BresenhamLineIterator & operator++()
    {
        // calculte the next pixel along the line and advance
        // ImageIterator accordingly
    }
    // ...

    bool operator==(BresenhamLineIterator const & r) const {
        return iterator_ == r.iterator_;  // delegate comparison to
    }                                     // internal image iterators

    ImageIterator::PixelType & operator*() {
        return *iterator_;  // likewise delegate access
    }

    ImageIterator iterator_;
    float x_, y_;
};
```

A general drawing algorithm would now look like this:

```
template <class Iterator, class Accessor, class Color>
void drawShape(Iterator i, Iterator end, Accessor a, Color color)
{
    for(; i != end; ++i)    a.set(color, i);
}
```

By passing different iterators, this algorithm can draw any kind of shape. Similarly, we can implement an algorithm to output the intensity profile along an arbitrary contour, and we can even apply the linear convolution defined here to any curved line, always using the same set of iterator adapters to determine the curve.

There are many more possible iterator adapters that we can not describe here. To mention just a few: chain code iterators proceed according to a given chain code. *Contour iterators* follow the contour of a labeled region in a segmented image (see Fig. 6.2). Region iterators scan all pixels in a given region, which lets us calculate region statistics or fill the region with a new color. Interpolating iterators can be placed on noninteger positions and return interpolated image data there.

6.8 Conclusions

This chapter introduced a generic programming approach to the development of reusable algorithms in image processing and analysis. It is based on ideas made popular by the C++ Standard Template Library and extends its abstract iterator design to the domain of 2-D images. This approach overcomes a number of limitations of traditional ways to design reusable software:

- Programming is more efficient than with traditional methods because a single algorithm implementation can be adapted to many different situations.
- There is only a small performance penalty compared to an optimized version of the algorithm. Once the optimizers catch up, it is expected that the performance penalty can be eliminated altogether.
- A smooth transition to generic programming is possible: iterators can be implemented for existing data structures. Switching to generic programming does not influence any existing code. Existing code can be transformed into a generic form as needed.
- Implementing generic algorithms is reminiscent of the traditional structured programming paradigm which with many researches in computer vision are already familiar.

Besides the results reported here, we have successfully applied the same techniques to build and manipulate graph-based representations of image contents, such as feature adjacency graphs, feature correspondences across image ssequences, scene graphs, and geometric models. Our experience also showed that a relatively small number of iterator, accessor and adapter implementations suffice to support a wide variety of different environments, image data types, and applications.

It has proven very effective to combine generic methods with other software design approaches such as component-based development (Chapter 4) and object-oriented design (Chapter 5). The basic idea here is to use objects and components to arrange functionality and to represent their relationships according to the needs of a particular application, while most of the real work is delegated to generic building blocks residing below. In this way, strengths and weaknesses of the different methods are optimally balanced.

For example, in a digital photogrammetry system under development in our institute, we have wrapped generic algorithms and data structures into an object-oriented class hierarchy. In contrast to earlier designs, a reorganization of this class hierarchy did not affect the algorithms' implementations—the compiler automatically generated the new versions needed by the revised objects, thus saving us from a lot of work. In another project, we similarly wrapped generic building blocks into PYTHON modules. PYTHON is a public domain scripting language that can be extended by C and C++ libraries (Website at `http://www.python.org`). Thus, with minimal additional effort we were able to build a simple (command driven) interactive image processing system on top of the generic VIGRA library.

Using the described approach, change and reuse of image processing and analysis algorithms has become much easier in our organization. Not only has programming productivity increased, but, perhaps more importantly, the psychological barrier to modify existing solutions has been lowered because it became so much easier and safer to adapt the existing code to new ideas. It would be highly desirable to officially standardize a common set of iterators and accessors in order to prevent any compatibility problems between implementations that may otherwise arise. For version 1.0 of the VIGRA library see `/software/06` on the CD-ROM. The latest version can be downloaded from `http://www.egd.igd.fhg.de/~ulli/vigra/`.

6.9 References

[1] Khoral Research Inc., (1996). KHOROS Documentation (http://www.khoros.unm.edu/).

[2] Musser, D. and Stepanov, A., (1989). Generic Programming. In *1st Intl. Joint Conf. of ISSAC-88 and AAEC-6*. Berlin: Springer.

[3] Musser, D. and Stepanov, A., (1994). Algorithm-oriented generic libraries. *Software - Practice and Experience*, **24(7)**:623–642.

[4] Musser, D. and Saini, A., (1996). *STL Tutorial and Reference Guide*. Reading, MA: Addison-Wesley.

[5] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., (1994). *Design Patterns*. Reading, MA: Addison-Wesley.

[6] Kühl, D. and Weihe, K., (1997). Data Access Templates, C++ Report July/August.

[7] Myers, N., (1995). A New and Useful Template Technique: Traits, C++ Report, June.

[8] Meyers, S., (1996). *More Effective C++*. Reading, MA: Addison-Wesley.

7 Application-oriented Assessment of Computer Vision Algorithms

Peter Klausmann[1], Stefan Fries[1], Dieter Willersinn[1], Uwe Stilla[2], and Ulrich Thönnessen[2]

[1]Fraunhofer-Institut für Informations- und Datenverarbeitung (IITB) Karlsruhe, Germany
[2]Forschungsinstitut für Informationsverarbeitung und Mustererkennung Ettlingen, Germany

7.1 Introduction

The crucial question to be answered during the assessment of a computer vision algorithm is to what extent is the algorithm performance useful? The utility of an algorithm can only be stated with respect to an application, hence the assessment of computer vision algorithms

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

is only possible if the application of the computer vision algorithm is given.

The assessment of a computer vision algorithm falls into two parts, namely, algorithm performance characterization, and the decision as to whether the attained performance is sufficient for the given application. For an application-oriented assessment of computer vision algorithms both questions must be addressed.

Section 7.2 presents an overview of approaches to characterize the performance of computer vision algorithms. Section 7.3 describes our concept for the application-oriented assessment of computer vision algorithms. The concept covers both the performance characterization and the assessment of algorithm performance. Our assessment system is described in Section 7.4. Section 7.5 describes the assessment of a detection algorithm for two different applications.

7.2 Analytical versus empirical performance analysis

Models to explain and to predict the performance of a computer vision system have gained increasing attention among the computer vision community during the last few years [1, 2, 3, 4, 5, 6]. Although there exist a considerable number of approaches in computer vision, the design of systems that are able to work accurately over a broad range of images is still difficult. Two mainstream approaches have been developed by researchers to characterize the performance of image processing algorithms. The first approach has concentrated on analytical investigation and sophisticated concepts of error propagation. The second approach has focused on the quantitative, empirical and experimental performance characterization.

7.2.1 Analytical performance characterization

In nearly all disciplines of image processing ideas have been proposed concerning algorithmic behavior under perturbation influence. In linear estimation a complete theory about error propagation in terms of covariance calculus exists [7]. Haralick [8] and Bar-Shalom and Li [9] generalized such approaches to nonlinear problems. Also, numerous analytical approaches have been proposed to investigate edge detectors [10, 11, 12, 13], which are a general front-end tool of image analysis systems. Texture recognition and image segmentation approaches have been evaluated analytically since the beginning of the 1980s [14, 15, 16].

Another major topic addressed by researchers is the control of random perturbations and the influence of clutter. They have attempted to relate the results of a vision system to image characteristics and to investigate the main influence factors on algorithm performance.

There are many reasons why the image processing methodology may fail [17, 18]: misinterpretations of data features caused by noise; inconsistent description of object features; missing contrast of the objects; imprecise edge detection; or background clutter. Therefore, the object hypotheses usually are not consistent with all data, not even with a subset for which an algorithm is trained. Thus, the characteristics of objects to be detected have been analyzed under different noise and clutter conditions and have been compared with the capacity of human observers. Several papers discuss the features of objects and background concerning the visibility to human observers. Conners and Ng [19] modeled the human preattentive vision system on the basis of gray-value co-occurrence matrices. Shirvaikar and Trivedi [20] used these matrices to describe the background clutter with a texture-based image clutter (TIC) measure and discussed the influence of this clutter measure on the detection performance and false alarm rates. Waldman et al. [21] suggested that spatial extent, contrast, clutter, movement, shape, number of targets and color play the main roles in performance prediction. Schmieder and Weathersby [22] first noticed the relevance of the relation between target and background in determining detection probability. They defined a signal-to-clutter ratio (SCR) and compared the detection capacity of human observers versus the SCR and several target resolutions. Ratches et al. [23] stated that the signal-to-noise-ratio (SNR) defined by the average contrast difference between target and background normalized by the average background intensity has a basic influence on the detection performance.

The main problem with analytical descriptions is that most vision systems consist of a combination of multitudinous steps. Each layer includes highly adapted parameters for a specific application task. The complexity of the computational procedure can make a complete analytical analysis infeasible. Additionally, both the applications as well as the different methodologies are hardly comparable and the disadvantage of one method is the main strength of another. Heuristic parts of the algorithm are specifically adapted to special situations and are seldom valid in a more general context. For this reason empirical performance characterization is to date a common way to answer the earlier-stated question, although an analytical answer would be highly desirable. Suggestions of how to treat the evaluation problem are discussed in the following.

7.2.2 Empirical performance characterization

An early comparative study of image processing algorithms is that reported by Weszka et al. [24]. They analyzed three feature-based texture-measure approaches and evaluated their performance using a fixed data set. Nowadays, it is still a common method to consider different sets of input images and to relate the performance of the algorithm to the variation of image and algorithm parameters. This topic is addressed by Kanungo et al. [25]. They started with a set of noise-free images and generated a large number of images from this base set by adding different realizations of noise and perturbations. They then developed operating characteristic curves of the algorithm errors as functions of signal-to-noise ratios and studied the perturbation influence on the algorithm performance. Haralick [26] provided some reference operating curves of false alarm rates versus the nondetection rate as a function of the number of samples and a prespecified error rate of the vision system. To formulate the evaluation problem in terms of empirical statistics, similar approaches have been employed by Bradley [27], Haralick [28], Liu et al. [29], Takeshita and Toriwaki [30], and Nyssen [31]. They reviewed the theory about receiver operating characteristics, hypothesis testing procedures and significance measures, providing additional background for statistical software validation. To treat procedures of considerable computational complexity modern resampling methods such as bootstrap have been used by Cho et al. [32], Jain et al. [33] and Chernick et al. [34]. To evaluate complex algorithms, Courtney et al. [35] suggested that each layer be modeled separately as probability density functions and then combined to predict detection and nondetection rates.

7.3 Application-oriented empirical algorithm assessment

The assessment concept presented in this section eventually yields a qualitative or quantitative statement of algorithm utility for a given application, or a ranking of several algorithms based on that statement of utility. The concept clearly distinguishes between the measurement of algorithm performance and the assessment of measured algorithm performance for a given application. Due to the lack of analytical models when predicting algorithm performance, care must be taken during the acquisition of test data in order to obtain relevant data that reflect the particular difficulties encountered during the prospective application. Another consequence of the stated lack of analytical models is that relevant data have to be provided to the developers so that they can adapt their algorithms to the particular difficulties of the application.

The subsequent assessment of the measured performance is based on a description of the requirements of the application, the *requirement profile*. The algorithm performance is compared to the requirement profile by means of the assessment function that is used to derive an application-specific *figure of merit* (FOM). This FOM can be used in two ways: first, to derive a statement as to whether or not the assessed algorithm is useful for the prospective application; and second, it may be used for ranking several algorithms according to their figure of merit. The last step of the assessment concept is to collect all of the information produced during the measurement and assessment procedure (e.g., performance profile, the assessment function, the analysis of single algorithm results, etc.) in an assessment profile that is presented to the prospective algorithm user as well as to the developer.

7.3.1 Task, performance criteria and requirement profile

A computer vision algorithm always performs its task in the context of an application [36]. Hence, the assessment of a computer vision algorithm can only be done with respect to the application. Further, it requires an agreement based on general criteria of engineering. This agreement must contain an explicit description of the algorithm task and has to define:

- parameters of the scene;
- objects of interest and their parameters;
- sensor type and sensor parameters;
- the expected computer vision results; and
- the prospective use of the computer vision results.

From the task definition *performance criteria* are derived. We distinguish between general criteria and criteria that are specific to the algorithm class. Examples for algorithm classes are detection, classification or parameter estimation. Typical performance measures for detection algorithms are, for example, detection rate and the false alarm rate. The performance of classification algorithms is usually described by confusion matrices, and parameter estimation algorithms can be characterized by estimation errors and standard deviations.

General criteria important for the user could be availability, reliability, sensitivity with respect to parameter variation, hardware requirements of the algorithm as well as resource consumption and execution time.

The intended use of the computer vision algorithm determines which criteria are of importance, and allows assigning required values to each of the selected criteria. Also, weighting factors (costs) for the selected criteria according to their importance for the application could be established.

The forementioned criteria and weights are fixed in the *requirement profile*. Together with a rule for combining the weighted criteria to obtain the figure of merit they define the so-called *assessment function*. The investigation of additional algorithm features (hardware requirements, etc.) would be described in the requirement profile as well.

7.3.2 Assessment data

This section describes *assessment data*, that is, all data that are necessary to perform the assessment of a computer vision algorithm: image data; collateral data; and truth data.

Type of image data. In order to verify an algorithm we distinguish between: (i) synthetic image data; (ii) image data of modeled objects; and (iii) image data of real objects [37].

Synthetic image data can be computed from scene descriptions by algorithms for computer graphics and virtual reality. There are many tools available to produce images by using a sensor model, an illumination model and an object or scene model. The advantage of using synthetic objects is the inherent knowledge about the truth of the modeled scene. This allows for a testbed structure for fully automatic testing [38].

With the growing demand for the photorealistic appearance of synthetic objects there has been a considerable increase in efforts to model and generate images. For some examinations image data of *model objects* offer an alternative [39]. The image can be taken regardless of the weather, at low cost and under reproducible conditions.

Working under changing conditions or using outdoor scenes the vision system has to cope with data that contain distortions and other perturbations from various sources. To date, it is not possible to realistically model all sources of perturbations that have an impact on the imaging process. For this reason the use of image data of *real objects* is a must for the assessment of a computer vision algorithm.

Collateral data. Besides the information given by the test images supplementary information may be available to the algorithm during operation. Such information includes geometric resolution, radiometric resolution, sensor geometry, and operational conditions. This information is called *collateral data*.

Truth data. The previous sections described the data available to the algorithm. For assessment purposes, there has to be a description of the scene represented by the image data as well. Figure 7.1 illustrates the data acquisition for the assessment of an algorithm that analyzes images of natural scenes. The scene description is referred to as *ground truth*.

It is important to mention here that the ground truth is application-dependent in the sense that it must contain that information that is relevant for the given application. In the case illustrated by Fig. 7.1, the relevant information consists of descriptions of the vehicles present in the scene.

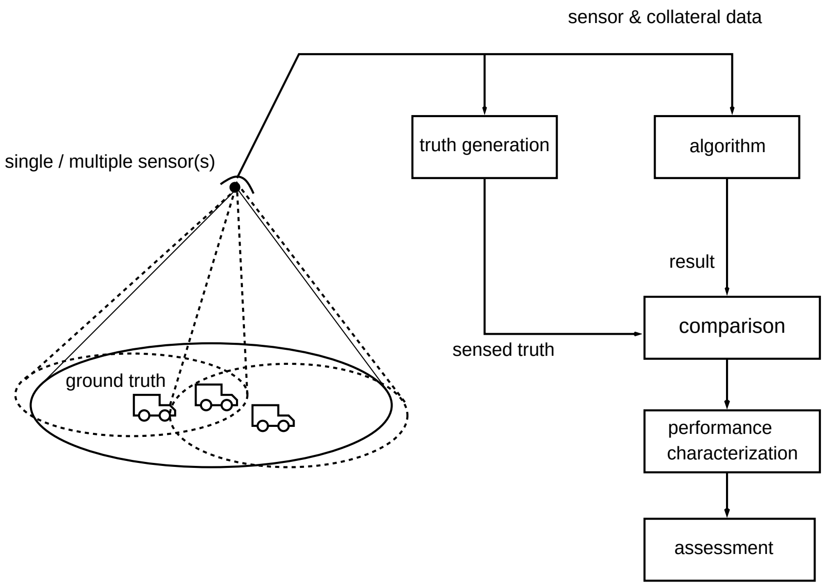


Figure 7.1: *Illustration of ground truth, sensed truth and algorithm assessment.*

Note that ground truth data may also describe objects that are invisible in the sensor images. One reason for this is occlusion, for example, cars occluded by trees. Another reason for objects missing in sensor images is illustrated in Fig. 7.1: the scene is captured by one or several sensors with different fields of view represented by three ellipses of different size. The sensor field of view depicted by a solid ellipse contains all objects present in the scene, while the two dashed ellipses contain only two objects each.

A problem that is not illustrated in Fig. 7.1 is due to perturbations that may cause algorithm failures. Knowledge about these perturbations is required for a detailed analysis of algorithm performance.

Because of the cited problems we assign a sensor-specific description to each image. This sensor-specific description is referred to as *sensed truth* in Fig. 7.1.

The sensed truth contains only those objects that are actually captured by the sensors, along with a list of potential sources of misinterpretations. The sensed truth represents the ideal result of a computer vision algorithm. In the philosophy of the assessment concept presented in this contribution, sensed truth has to be generated interactively by a human interpreter. In order to establish the list of potential sources of misinterpretations, a verification in the scene may be required after a check of the acquired images.

7.3.3 Image data acquisition

The algorithm performance in general is not independent of scene parameters and parameters of the image acquisition process. For this reason the images used for performance characterization must be representative for the application if the measured performance is required to be representative. The problem that arises involves the number of parameters of the imaging process. Relevant parameters are, for example, scene and object parameters, parameters of sensor and sensor carrier, time of day, time of year and weather conditions. Generally, the space spanned by the relevant parameters is too large to be covered entirely by the test images. In practice a compromise must be made to concentrate on the most important parameters regarding the intended application and to acquire a subset of all possible images.

The data acquisition may be expensive and the generation of truth is time consuming. As empirical tests are expensive joint tests are necessary. They allow the exploitation of the resources of several institutions, academia and industry, in order to define and perform the tests, including the preparation of truth, the necessary calibration of the system, the huge amount of repeated measurements, and the proper analysis [2]. These are some of the advantages of establishing an assessment center.

7.3.4 Provision of training data for development

The stated lack of formal models for the imaging process implies that the specification of a computer vision algorithm cannot be done formally, but must be completed by real images of application-relevant scenes. For this reason the relevant data set is split into a test data set and a training data set. The training data set must be representative of the entire data set. Using this data set, the software developer may adapt his or her algorithm in order to obtain an optimal result for the given application.

7.3.5 Algorithm performance characterization

The steps preceding performance characterization are shown in Fig. 7.1. The algorithm is run on the images (sensor data) whereas for each image the collateral data are made available to the algorithm as well.

The succeeding step is the comparison between algorithm results and sensed truth descriptions. For this comparison, a similarity metric has to be defined. Depending on the type of information that the algorithm has to extract from the image (algorithm class), different similarity measures may be appropriate:

- result point inside/outside truth polygon;

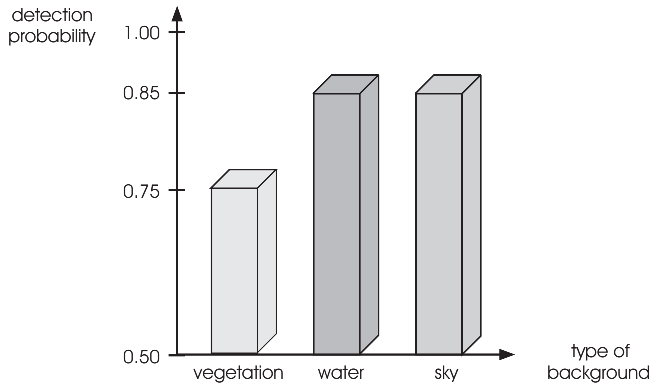


Figure 7.2: *Example of a performance profile.*

- correlation coefficient;
- sum of squared differences;
- contour difference; or
- structural matching distances, etc.

The computed differences and distances between algorithm results and sensed truth are then processed to obtain the actual algorithm performance criteria.

For a detailed insight we represent the algorithm performance over the space that is spanned by the image acquisition parameters and scene parameters such as background, object type, and movement, etc. This representation is called the *performance profile* of the algorithm. An example for the performance profile of a detection algorithm is the representation of its detection probability (performance criterion) over object background (scene parameter) (see Fig. 7.2).

The performance profile is an important piece of information for the user of the computer vision algorithm because it contains detailed information about the expected algorithm performance under different operational conditions.

7.3.6 The assessment function

This section describes how an application-specific figure of merit can be obtained from the performance profile consisting of the measured performance criteria $k_1, k_2, \ldots, k_n$. The computation is done by means of the assessment function described in the requirement profile. As the requirement profile also defines some required values for the performance criteria, the FOM could be regarded as a measure of how well the algorithm performance satisfies the requirements of a given application.

Formally, the FOM is the result of applying the *assessment function* using the measured performance criteria as parameters

$$\mathrm{FOM} = f(k_1, k_2, \ldots, k_n) \tag{7.1}$$

In a simple approach, an assessment function could weigh each criterion k_i according to its costs C_i defined in the requirement profile and the FOM would be defined by summing the weighted criteria

$$\mathrm{FOM} = \sum_{i=1}^{n} C_i \, k_i \tag{7.2}$$

Note that the FOM may also be represented as a function of the algorithm parameters that have an impact on the algorithm performance. As the assessment function is part of the requirement profile and as the obtained FOM is crucial for the decision regarding algorithm utility, it has to be stressed again that the requirement profile needs an agreement among user, developer and assessing institution.

7.3.7 Overview of the assessment procedure

Algorithm developers and users together with the assessing institution specify the application task from which an assessment task has to be derived. Figure 7.3 outlines the information flow involved in the assessment of a computer vision algorithm.

First, the application is analyzed to obtain relevant *scene and acquisition parameters*, and to define the *performance criteria* together with their *weightings* that determine the *assessment function*. A comprehensive assessment requires a complete acquisition of the scene parameters and their variation. However, the specific combinatorial relationships usually do not demand the processing of all variations. Rather, the strategy must be to restrict the assessment to the most relevant cases. The set of scene and acquisition parameters specifies the procedure of *data acquisition*. The resulting *assessment data* consist of the images taken by the sensor(s), the corresponding collateral data, and the *sensed truth* that has to be generated application-specific.

The *training data* (a small but representative subset of the assessment data) are delivered to the developer for *algorithm training*. After training the algorithms they are run on the remaining assessment data, called the *test data*. It is worth mentioning that during the algorithm test the algorithms have access only to image and collateral data but not, of course, to the truth data (*truth data extraction*).

Afterwards, the *algorithm result* is compared to the ideal result (the *truth*) by performing a *distance measurement*. A subsequent *statistical analysis* yields the *performance profile* of the algorithm. During

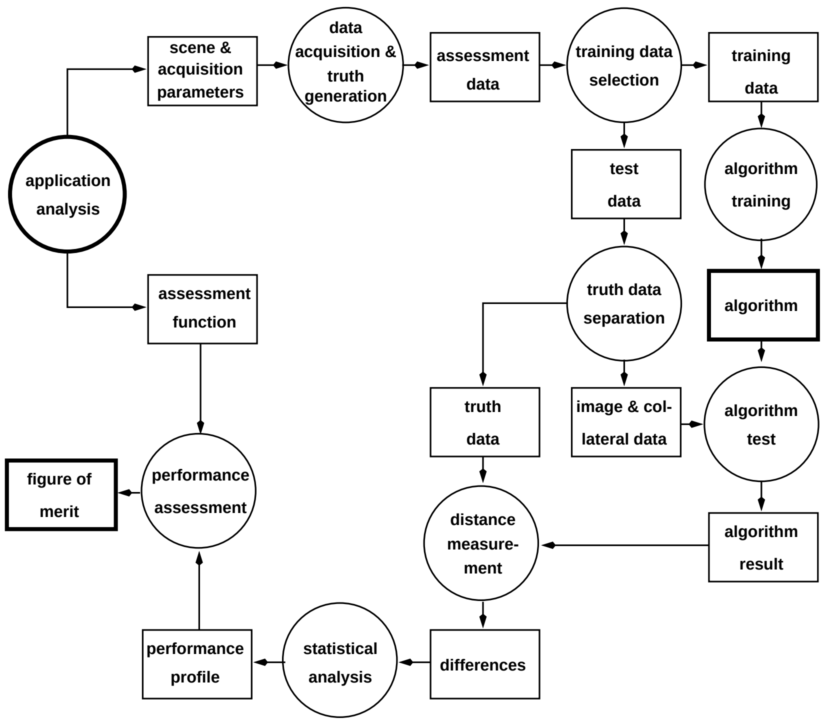
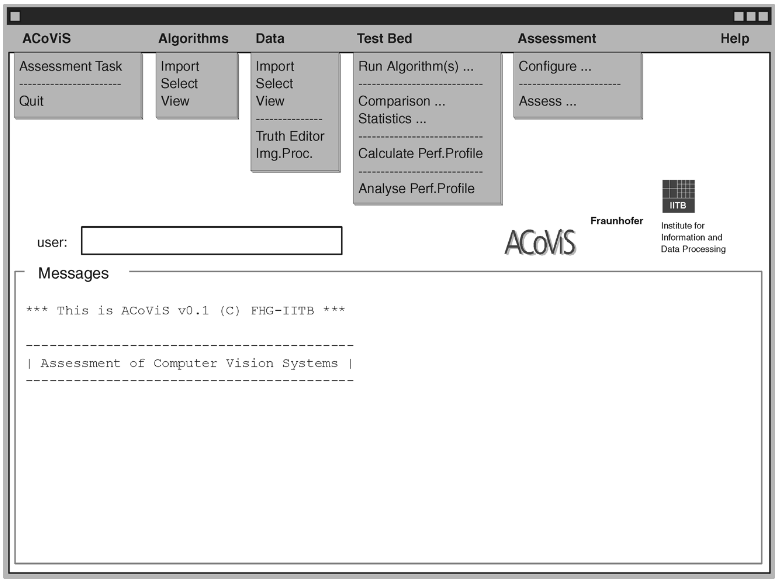


Figure 7.3: *Overview of steps involved in the assessment procedure. Boxes represent data, circles represent processes. The assessment of an algorithm begins with the application analysis and results in the figure of merit (FOM). These three major elements in the assessment procedure are highlighted.*

the *performance assessment* the application-specific FOM is computed from the algorithm performance profile by means of the assessment function. In the frame of an assessment strategy several competing algorithms may be evaluated and the best one is chosen for the application on the basis of the FOM.

7.4 The assessment system

Some, but not all steps of algorithm assessment can be executed automatically by means of a software tool collection [40, 41]. For example, the application of an algorithm to the test data would be a tideous task without the help of such a software. Likewise, managing large test data sets can essentially be simplified by a tool that provides management functionality in combination with a database. Hence, a toolbox was developed to support algorithm analysis and assessment, an overview of which will be given in this section.



```
*** This is ACoViS v0.1 (C) FHG-IITB ***

----------------------------------------
| Assessment of Computer Vision Systems |
----------------------------------------
```

Figure 7.4: *The graphical user interface of the assessment system with menues pulled down.*

Revising the concept described in the foregoing sections one may identify five main topics that have to be addressed by the assessment system:

- control and configuration of the assessment task;
- management of test data;
- management of tested algorithms;
- performance measurement (*test bed*); and
- performance assessment.

It was a major goal of the design stage to reflect these steps on the top of the assessment system, which resulted in the graphical user interface shown in Fig. 7.4. The underlying functionality that is accessible through the menu entries will be outlined in the following.

Assessment Task. As stated in the foregoing, the main viewpoint for the assessment of computer vision algorithms is the intended use of an algorithm. The application essentially determines the parameters of the assessment, for example, which algorithms are available for the given application, which images are available and how the algorithm performance is to be assessed. Together with the requirement profile the application has to be formalized as an *assessment task* and committed to the system. This is accomplished by means of an input sheet where all of the relevant parameters are to be stated.

Data management. An essential part of a software system for algorithm testing is the management of data. There are two aspects to mention: first, the large number of images necessary to yield statistically significant performance measures; and second, the variety of the images concerning their content and formation (sensor, scene, mapped objects, etc.) required for differerent applications. Especially the latter is crucial for data management: Every image should be stored together with a description that allows a selection of images according to some properties relevant for a given application. Finally, sensed truth has to be provided for every image as well.

The following three functions of the assessment system are available for data management: *import* of newly acquired data, *selection* of data to accomplish a given assessment task; and statistical analysis (*view*) of test data in order to check for statistically significant occurrences of application-relevant image content.

Algorithm management. For the management of algorithms a function comparable to that of data management is provided. First, an *import* tool provides for the gathering of an algorithm together with its specification and parameters. Next, the *selection* of algorithm(s) to be tested is carried out according to a given application. A tool to *view* the specification of a stored algorithm provides detailed information about the algorithm properties.

Test bed. The steps that are supported by the test bed of the assessment system are drafted in Fig. 7.3: execution of the algorithm using the available test data; comparison between algorithm results and truth, using an appropriate distance measure; computation and analysis of algorithm performance profiles.

Assessment. The crucial step of the assessment concept described in this contribution is the *assessment* of the algorithm performance regarding the underlying application. Two tools are provided for editing an assessment function and to perform the assessment and to display the results.

7.5 Example: Assessment of a detection algorithm

This section shows how to assess a vehicle-detection algorithm in the context of two different applications. For both applications the algorithm task consists in detecting vehicles in infrared imagery.

The considered algorithm returns the image coordinates of the centroid of detected vehicles. Additionally, for each centroid the algorithm calculates a number indicating its own confidence into the produced result. This number is called the *score* of the algorithm result.

The truth for each vehicle is given by a surrounding polygon in image coordinates. The comparison between algorithm results and truth polygons consists in checking if the result centroids are inside the corresponding truth polygons. The following cases may occur:

- One and only one result centroid is in a truth polygon (*detection*);
- More than one result centroid is in a truth polygon (*multiple detection*);
- A truth polygon exists without a corresponding result centroid (*miss, nondetection*);
- A result centroid is outside of all truth polygons (*false alarm*); and
- No result centroid is generated by the algorithm given an image without any truth polygon (*background*).

For each case the corresponding *rate* is calculated by normalizing the number of occurrences. In this example, the detection rate, nondetection rate and the multiple detection rate were calculated by normalizing the corresponding numbers by the number of objects. The background rate is calculated by normalization to the number of images and the false alarm rate by normalization to the number of events, which is the sum of all detections, multiple detections, misses, false alarms and backgrounds.

To gain insight into the algorithm behavior with respect to the score threshold, one can calculate the mentioned rates while ignoring events whose scores are smaller than a given *score threshold*. By varying the score threshold, one obtains a plot similar to the one in Fig. 7.5, which shows the detection rate vs the score threshold.

One major problem is how to build from these simple performance measures a FOM indicating the utility of the algorithm with regard to the application. This was done by the construction of an assessment function R working as a criterion that has to be optimized by variation of the score threshold s

$$R\,(d, f, n, b, m; s) = C_d\, d(s) + C_f\, f(s) + C_n\, n(s) + C_b\, b(s) + C_m\, m(s) \tag{7.3}$$

where $C_i \in [-1, 1]$, $i \in \{d, f, n, b, m\}$, and $d(s)$ is the detection rate, $f(s)$ is the false alarm rate, $n(s)$ is the nondetection rate, $b(s)$ is the background rate, and $m(s)$ is the multiple detection rate.

The constants C_i are application-dependent *cost* factors, which are used to address the relative importance of each event-type. A negative cost factor indicates a benefit, and a positive cost factor indicates a

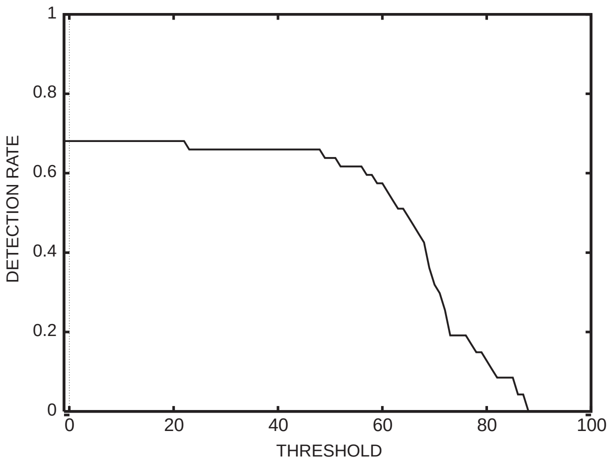


Figure 7.5: *Detection rate over the score threshold s.*

penalty. In respect thereof, $R(d, f, n, b, m; s)$ can be interpreted as the *risk* of applying the algorithm while rejecting results with score values smaller than s. In the underlying example, the FOM is defined as follows:

$$\text{FOM} = \min_s R(d, f, n, b, m; s) \tag{7.4}$$

It takes into account the weighted contribution of each performance criteria over the range of the score threshold s (see Fig. 7.6) and seeks the result with the lowest risk.

Using qualitative cost factors one can illustrate the assessment principle. The two applications considered here are:

- an *information gathering system (IGS)*), in which the algorithm results are further processed by human operators; and
- an *autonomous system (AS)*, in which decisions of the algorithm are directly taken without human interaction.

The cost factors are determined by applying the following considerations:

- detections/background: correct decisions that are rewarded in both applications (negative costs, cost factor -1.0);
- definitely, nondetections are counterproductive to the algorithm task. Therefore, they are penalized by a cost factor of 1.0 for both applications;
- multiple detections may lead to an incorrect evaluation of the observed situation in the IGS. Therefore the multiple detection rate is weighted with a cost factor of 0.7; for an AS the crucial information, for example, for obstacle avoidance, is the position of detected

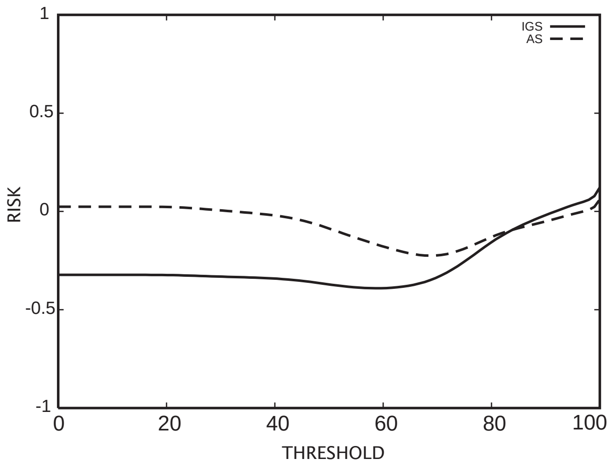


Figure 7.6: *Assessment function of the information gathering system (IGS, solid line) and the autonomous system (AS, dotted line). The minimum indicates the FOM and the task-specific operation point.*

objects. Multiple detection hypotheses are less critical for this application because they do not lead to incorrect decisions. The cost factor is therefore chosen to be 0.1;

- false alarms may lead to incorrect decisions in the AS and are therefore penalized by a factor of 1.0. In the interactive application a false alarm can be rejected by a human operator and thus is less problematic. The cost factor is chosen to be 0.5.

As already mentioned in the foregoing, the threshold applied to the scores of the detection hypotheses is used to reject weak algorithm results. The impact thereof on the risk depends on whether the rejected result is a detection, a multiple detection, a miss, a false alarm, or a background. For instance, the rejection of multiple detections and false alarms reduces the risk, whereas rejected detections and backgrounds increase the overall risk of the detection algorithm.

Figure 7.6 shows a plot of the risk $R(d, f, n, b, m, s)$ for both applications over the variation of the corresponding score threshold. The IGS has a minimum risk at a threshold of 60 indicating the optimal operation point. By contrast, the optimal operation point for the autonomous system is achieved at a score threshold of 70.

Further, the minimum risk of using the algorithm in the interactive application less than the minimum risk of using it in the autonomous application, which means that the algorithm has a higher utility for the IGS than for the autonomous system.

7.6 Conclusion

The assessment of computer vision algorithms is more than just a question of statistical analysis of algorithm results. Rather, the algorithm field of application has to be taken into account as well. Assessment as understood in this contribution quantifies the utility of image analysis algorithms with respect to given applications. A clear distinction between performance analysis and performance assessment has to be made. The underlying algorithm performance is determined experimentally. The final statement, how much an algorithm fulfills its requirements, demands basic knowledge about its application-dependent context. The connection between algorithm performance and this context is modeled by an application-dependent assessment function.

Acknowledgments

This work was funded by the German Federal Office for Defense Technology and Procurement under Project E/F41B/V0037/Q5242. The authors wish to thank B. Dürr, Dr. U. Jäger, and G. Saur from the Fraunhofer-Institut für Informations- und Datenverarbeitung, Karlsruhe, Germany for many helpful discussions and suggestions. The detection algorithm was developed by M. Müller, L. Berger, M. Esswein, N. Heinze, Dr. W. Schwerdtmann, and C. Sung (all from the Fraunhofer-Institut für Informations- und Datenverarbeitung). G. Hofele from the Forschungsinstitut für Informationsverarbeitung und Mustererkennung, Ettlingen, Germany provided the data required for the assessment example.

7.7 References

[1] Förstner, W., (1994). Diagnostics and performance evaluation in computer vision. In *Proceedings of the Workshop on Robust Computer Vision*. Seattle, USA.

[2] Förstner, W., (1996). 10 Pros and cons against performance characterisation of vision algorithms. In *Proceedings of the ECCV Workshop on Performance Characteristics of Vision Algorithms*. Cambridge, UK. http://www.vision.auc.dk/ hic/perf-proc.html.

[3] Haralick, R., (1994). Overview: Computer vision performance characterization. In *Proceedings of the ARPA Image Understanding Workshop*, pp. 663–665. Monterey.

[4] Haralick, R. M., (1994). Performance characterisation protocol in computer vision. In *Proceedings of the ARPA Image Understanding Workshop*, pp. 667–673. Monterey.

[5] Jain, R. C. and Binford, T. O., (1991). Ignorance, myopia, and maiveté in computer vision. *Computer Vision, Graphics, and Image Processing: Image Understanding*, **53**(1):112–117.

[6] Zamperoni, P., (1996). Plus ça va, moins ça va. *Pattern Recognition Letters*, **17**:671–677.

[7] Förstner, W., (1987). Reliability analysis of parameter estimation in linear models with applications to mensuration problems in computer vision. *Computer Vision Graphics and Image Processing*, **40**(3):273–310.

[8] Haralick, R. M., (1994). Propagating covariances in computer vision. In *Proceedings of the International Conference on Pattern Recognition*, pp. 493–498. Jerusalem, Israel.

[9] Bar-Shalom, Y. and Li, X., (1993). *Estimation and Tracking*. Boston: Artech House.

[10] Demigny, D. and Kamlé, T., (1997). A discrete expression of Canny's criteria for step edge detector evaluation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **19**.

[11] Fuchs, C., Lang, F., and Förstner, W., (1997). On the noise and scale behaviour of relational descriptions. In *DAGM-Workshop on Performance Characteristics and Quality of Computer Vision Algorithms*. Braunschweig, Germany.

[12] Salotti, M. and Garbay, C., (1996). Evaluation of edge detectors: critics and proposal. In *Proceedings of the ECCV Workshop on Performance Characteristics of Vision Algorithms*. Cambridge, UK. http://www.vision.auc.dk/˜hic/perf-proc.html.

[13] Spreeuwers, L. and van der Heijden, F., (1992). Evaluation of edge detectors using average risk. In *Int. Conf. on Pattern Recognition (ICPR92)*, Vol. 3, pp. 771–774. The Hague.

[14] Conners, R. W. and Harlow, C. A., (1980). A theoretical comparison of texture algorithms. *IEEE Trans. Pattern Recognition and Machine Intelligence*, **2**(3):204–222.

[15] de Graaf, C., Koster, A., Vincken, K., and Viergever, M., (1992). Task-directed evaluation of image segmentation methods. In *Proc. Int. Conf. Pattern Recognition*, Vol. 3, pp. 219–222. The Hague.

[16] Wang, Z., Guerriero, A., and De Sario, M., (1996). Comparison of several approaches for the segmentation of texture images. *Pattern Recognition Letters*, **17**:509–521.

[17] Lindenbaum, M., (1997). An integrated model for evaluating the amount of data required for reliable recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence Systems*, **19**(11):1251–1264.

[18] Sarachick, K., (1994). The effect of Gaussian error in object recognition. In *Image Understanding Workshop*, Vol. 2. Hyatt Regency Hotel, Monterey, California.

[19] Conners, R. and Ng, C., (1989). Developing a quantitative model of human preattentive vision. *IEEE Trans. Systems, Man, and Cybernetics*, **19**(6):204–222.

[20] Shirvaikar, M. and Trivedi, M., (1992). Developing texture-based image clutter measures for object detection. *Optical Engineering*, **31**(12):2628–2639.

[21] Waldman, G., Wootton, J., Hobson, G., and Luetkemeyer, K., (1988). A normalized clutter measure for images. *Computer Vision, Graphics, and Image Processing*, **42**:137–156.

[22] Schmieder, D. and Weathersby, M., (1983). Detection performance in clutter with variable resolution. *IEEE Trans. Aerospace and Electronics Systems*, **19**(4):622–630.

[23] Ratches, J., Walters, C., Buser, R., and Guenther, B., (1997). Aided and automatic target recognition based upon sensory inputs from image forming systems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **19**(9): 1004–1019.

[24] Weszka, J., Dyer, C., and Rosenfeld, A., (1976). A comparative study of texture measures for terrain classification. *IEEE Trans. Systems, Man and Cybernetics*, **6**:269–285.

[25] Kanungo, T., Jaisimha, M., Palmer, J., and Haralick, R. M., (1995). A methodology for quantitative performance evaluation of detection algorithms. *IEEE Trans. Image Processing*, **4**(12):1667–74.

[26] Haralick, R., (1989). Performance assessment of near-perfect machines. *Machine Vision and Applications*, **2**:1–16.

[27] Bradley, A. P., (1996). ROC curves and the χ^2 test. *Pattern Recognition Letters*, **17**:287–294.

[28] Haralick, R., (1996). Detection performance methodology. In *ARPA Image Understanding Workshop*, pp. 981–983. Palm Springs, USA.

[29] Liu, X., Kanungo, T., and Haralick, R., (1996). Statistical validation of computer vision software. In *ARPA Image Understanding Workshop*, pp. 1533–1540. Palm Springs.

[30] Takeshita, T. and Toriwaki, J., (1995). Experimental study of performance of pattern classifiers and the size of design samples. *Pattern Recognition Letters*, **16**:307–312.

[31] Nyssen, E., (1996). Evaluation of pattern classifiers—testing the significance of classification using an exact probability technique. *Pattern Recognition Letters*, **17**:1125–1129.

[32] Cho, K., Meer, P., and Cabrera, J., (1997). Performance assessment through bootstrap. *IEEE Trans. Pattern Analysis and Machine Intelligence Systems*, **19**(11).

[33] Jain, A. K., Dubes, R. C., and Chen, C.-C., (1987). Bootstrap techniques for error estimation. *IEEE Trans. Pattern Recognition and Machine Intelligence*, **9**(5):628–633.

[34] Chernick, M., Murthy, V., and Nealy, C., (1985). Application of bootstrap and other resampling techniques: Evaluation of classifier performance. *Pattern Recognition Letters*, **3**(3):167–178.

[35] Courtney, P., Thacker, N., and Clark, A., (1996). Algorithmic modelling for performance evaluation. In *Proceedings of the ECCV Workshop on Performance Characteristics of Vision Algorithms.* Cambridge, UK. http://www.vision.auc.dk/ hic/perf-proc.html.

[36] Rivlin, E., Aloimonos, Y., and Rosenfeld, A., (1991). *Purposive Recognition: A Framework.* Technical Report CAR-TR-597, Center for Automation Research, University of Maryland.

[37] Stilla, U., (1997). Automatische Extraktion von Gebäuden aus Luftbildern mit Produktionsnetzen. In *DGPF-Jahrestagung, Arbeitskreis Bildanalyse.*

[38] Stilla, U., Michaelsen, E., and Lütjen, K., (1996). Automatic extraction of buildings from aerial images. In *Mapping Buildings, Roads and Other Man-Made Structures from Images*, F. Leberl, R. Kalliany, and M. Gruber, eds. Wien: Oldenburg.

[39] Huertas, A., Bejanin, M., and Nevatia, R., (1995). Model registration and validation. In *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, A. Gruen, O. Kuebler, and P. Agouris, eds., pp. 33–42. Basel.

[40] Clark, A. F., (1996). HATE: A Harness for Algorithm Testing and Evaluation. Available from http://peipa.essex.ac.uk/hate/.

[41] De Boer, C. and Smeulders, A., (1996). BESSI: An experimentation system for vision module evaluation. In *International Conference Pattern Recognition ICPR96*, p. C70.2. Vienna, Austria.

8 A Hybrid Neuro-Artificial Intelligence-Architecture

Georg Hartmann, Ulrich Büker, and Siegbert Drüe

Heinz Nixdorf Institut, Universität Paderborn, Germany

8.1 Introduction

Many artificial vision systems have been developed in recent years and their performance is steadily improving. Compared with the performance of biological visual systems, however, the features of artificial systems are rather modest. Our human visual system is impressive by

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

the almost unlimited number of recognition problems that we can easily solve without being conscious of the difficulty. We can rapidly learn objects without training sequences, and we are able to recognize them robustly "at a glance." We are not even conscious that there could be problems with invariances or 3-D vision. Without the use of any zoom lens we are able to get a panoramic view of an extended scene, and somewhat later we can concentrate on a small detail in highest resolution. We can actively and selectively examine characteristic details that we expect on the basis of previously acquired knowledge.

We are still far from understanding the human brain and its architecture, and we feel it is useless to attempt to imitate it. On the other hand, the observed performance of our visual system shows that there exist solutions for all the forementioned features. In our opinion it is very stimulating to have these outstanding features and to aim at a technical solution. The development of our system was guided by finding general solutions and by discarding all solutions limited to special fields of application. Thus the presentation of our system in this chapter will give more than a description of the architecture and of its performance. We shall also try to show the arguments and the conclusions behind our decisions for building our system in this way.

Our goal to come as close as possible to the features of biological systems led us to very specific technical solutions. We felt free to compare these solutions with knowledge about biological solutions, and in many cases we found surprising similarities. For this reason we sometimes use a biological terminology to simplify explanations.

We begin with the observation that human visual behavior shows fast learning and quick, holistic recognition of objects "at a glance." Holistic recognition provides all invariances and so we discuss different invariance strategies in Section 8.2.1. We select a mechanism working under the constraints of pattern matching and fast learning. In Section 8.2.2 the underlying normalization mechanisms are discussed. Our technical solution requires an inhomogeneous retina and spatial frequency channels well known from biology. Normalized representations can be learned holistically, however, neither estimation of normalization parameters nor fixation of objects can be achieved without inevitable errors. These errors lead to the separability-tolerance problem discussed in Section 8.2.3.

Tolerant contour representations providing a way out of this dilemma are introduced in Section 8.2.4. The algorithm providing tolerant contour representations allows a neural interpretation that shows surprising similarities with cortical representations by complex cells. In Section 8.2.5 we explain the reason why holistic learning of tolerant representations needs no training sequences. This is due to shifting generalization away from the learning network towards the tolerance of representations.

Next, we ease some restrictions and introduce color representations in Section 8.3.1. We also expand our system to view-based 3-D recognition in Section 8.3.2, and we can show that without any change our representations also exhibit tolerance against minor perspective distortions. As a result we need no special module interpolating between learned views.

We also have replaced our simple binary contour detectors that were initially used in our system. In Section 8.3.3 we introduce tolerant Gabor-based representations and improved matching strategies. The benefits of Gabor-based representations in real world scenes with low contrast or blurred edges are evident.

However, under these circumstances segmentation and foveation of objects becomes difficult or even impossible. Therefore, we introduced a strategy of extrafoveal recognition in Section 8.3.4 that is also known from our visual experience. In Section 8.3.5 we show two crucial limitations of our holistic system: there is no chance of recognizing heavily occluded objects, and due to the tolerance details beyond a limit can not be discriminated.

These limitations are removed by combining our holistic system with an artificial intelligence (AI) system. The idea again results from observations of human visual behavior. We sequentially recognize occluded objects from partial views and their relations, and we sequentially go to details of an object by shrinking the visual field. After a discussion of the limits of classical AI systems in Section 8.4.1 we can show the benefits of a hybrid system in Section 8.4.2. Modeling with holistically learned partial views provides rich features, easy modeling, and a dramatically reduced search space.

Obviously, available AI shells must be modified for modeling with holistically learned features, and appropriate instantiation strategies are shown in Section 8.4.3. Some results with overlapping 2-D objects and with modeling of 3-D objects are shown in Section 8.4.4. We also provide the first results with automatic generation of AI models in Section 8.4.5.

Finally, we would like to point out why we drop the usual discussion of references in the introducing chapter. This chapter covers a very wide field, from biology to AI, from low-level vision to learning, from color to 3-D. A careful discussion would inflate this introduction, and so we give the references at the appropriate locations. Moreover, there is already a chapter (Chapter 9) in this handbook that gives an overview of active vision systems.

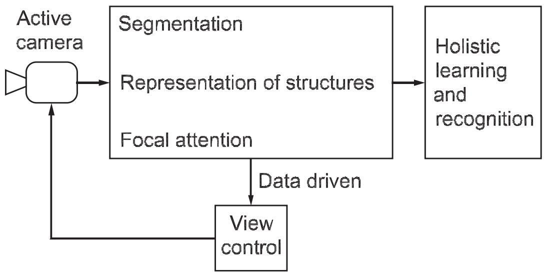


Figure 8.1: Implicit generation of invariances by generalization in a neural network.

8.2 Holistic learning and recognition of segmented two-dimensional objects

8.2.1 Invariances

Observation of human visual behavior shows fast learning and quick, holistic recognition of objects "at a glance." Objects of low and medium complexity such as traffic signs, tools, written syllables, etc., are immediately recognized. On the other hand, view sequences are observed if objects or scenes of higher complexity are analyzed in detail by looking at different parts of scenes or at object components. While sequential analysis may be guided by knowledge, holistic recognition seems to be based on matching with previously learned patterns.

The very first idea is obviously to realize a holistic system by putting a neural network onto a neural representation of the object (Fig. 8.1). In this widespread approach invariances are to be provided implicitly by the neural network. However, there are some major disadvantages with this approach. If invariances are implicitly generated, this is always due to generalization, and generalization can only be achieved on the base of extended training sequences. This would be in contradiction to our goal of rapid learning as it is observed in human vision. If we see the face of an unknown person only for a short time we will easily recognize the person in a new situation at a different distance without having learned the face at all distances in a training sequence. But also if we would accept training sequences in a technical system, there remains the disadvantage of limited performance. Of course, well-trained networks tolerate slightly different size and orientation of objects, for example, of handwritten figures. Invariances over a wide range of distances and orientations, however, are hardly provided implicitly by sheer training of networks. But even if there exists some kind of omnipotent network generalizing the image of an object independent of its size and orientation, there would be an inevitable loss of information. The system would not be able to distinguish between the figures *9* and *6*, or between the characters n and u as they are only distinguished by a rotation from each other. Similarly, the system could no longer distinguish between a car and a Matchbox car. If invariant features are extracted and learned these problems remain, while the new problem arises that invariant features are frequently domain-specific. Finally, providing invariances by AI-modeling is inherently sequential and is not helpful in a holistic approach.

Explicit generation of invariances based on parametric mappings, however, seems to be a mechanism compatible with a holistic system. (Appropriate implementations are described in more detail in the next chapter.) The general idea is to scale and to rotate the image $\boldsymbol{I}$ in such a way that the object always seems to be at a fixed distance r_0 and in a fixed orientation α_0. This is achieved by a scaling operation $\mathcal{S}(r)$ depending on the estimated object distance r, and by a rotation $\mathcal{R}(\alpha)$ depending on the estimated orientation α. As $\mathcal{S}(r)$ makes the normalized image independent from distance, an object (e. g., a car) can be learned once at an arbitrary distance and will be recognized immediately at any distance. A differently sized similar object (e. g., a Matchbox car) will always be mapped onto a different (smaller) normalized image, and thus it will never be confused with a car. Accordingly, $\mathcal{S}(r)$ provides distance invariance that is obviously more useful than size invariance. Of course, the mapping $\mathcal{S}(r)$ can be influenced deliberately, and by adjusting the scale factor, a car can be identified to be similar to a Matchbox car. The rotation operation $\mathcal{R}(\alpha)$ makes the normalized image independent of the object orientation and so, for example, a tool can be learned once in an arbitrary orientation and will be recognized immediately in any orientation. Orientation invariance, however, is provided by $\mathcal{R}(\alpha)$ explicitly and so this mapping can be deliberately switched on or off. In the case of reading characters the mapping will be switched off and the system will be able to distinguish between the figures *9* and *6*, or between the characters n and u. In conclusion, the parametric mappings $\mathcal{S}(r)$ and $\mathcal{R}(\alpha)$ together with an active camera provide distance, orientation, and shift invariance, which is sufficient for our first goal, for holistic learning and recognition of segmented 2-D objects. However, we will see that the special implementation of these invariances will also provide a solid basis for invariance with respect to perspective distortion.

8.2.2 Building normalized representations

In a world of continuous mathematics the implementation of the forementioned mappings is trivial. However, in the real world of sampled images we have to deal with limited resolution of the pixel matrix at the input of the mappings. On the other hand, the feature vector at the output should have a fixed number of components independent of scaling and rotation. Thus, what we would like to have for the mapping $S(r)$ is some kind of rubber-sheet retina that can be shrunk or stretched to the actual size of $\boldsymbol{I}$. On this retina, images of different size would always be represented by a constant number of pixels and there would not be any problem with a constant number of components in a feature vector. Zoom lenses can partially solve this problem in a technical vision system (see Section 8.3.3). However, they are not available in the biological vision system. When we started to investigate the constraints of a system without zoom lenses, we did not look at the biological system in order to make a copy of it. In fact, we went in another direction: we found a way to implement all the necessary features of a normalizing mapping and were surprised to find these construction principles also in the visual system.

Independent of the implementation of the normalizing mapping an inhomogeneous sampling proves to be optimal. This is easily demonstrated: we choose a maximum resolution at which an object, for example, a car shall be represented if its image covers the whole retina. Now we place the car at twice the distance, and represent it at the same resolution in the center of the visual field. Then the pixel distance in the central part must not exceed half the distance of the peripheral part. If the car is at four times distance, the pixel distance in the quarter-sized central region must not exceed a quarter of the distance in the periphery, and so on. If we stop this procedure at a given distance we see an increasing pixel density from the periphery to the center but no singularity. This density distribution is well known in vertebrate retinae. It fulfills our condition of constant resolution independent of distance with a minimum number of pixels (or fibers of the optical nerve in biology).

This observation immediately suggests a system based on a log-polar retina (compare also Volume 2, Section 3.1.4). Scaling and rotation are easily implemented by shifts in radial or tangential direction or by presynaptic switching in a biological system [1]. However, there are two crucial disadvantages. On the one hand, there is a limited separability due to the under-representation of peripheral parts of objects that are essential for shape description. On the other hand, there is a high sensitivity against minor object shifts due to an over-representation of the central part. All attempts to increase shape description by increasing overall resolution makes the over-representation of the central part still worse, and vice versa (Fig. 8.2).

The only solution keeping the advantages of the log-polar retina and avoiding its disadvantages is a representation of the image in a set of log-polar retinae of different resolution (spatial frequency channels). We have implemented a normalizing mapping based on this concept and we will now give a more formalized description of our solution.

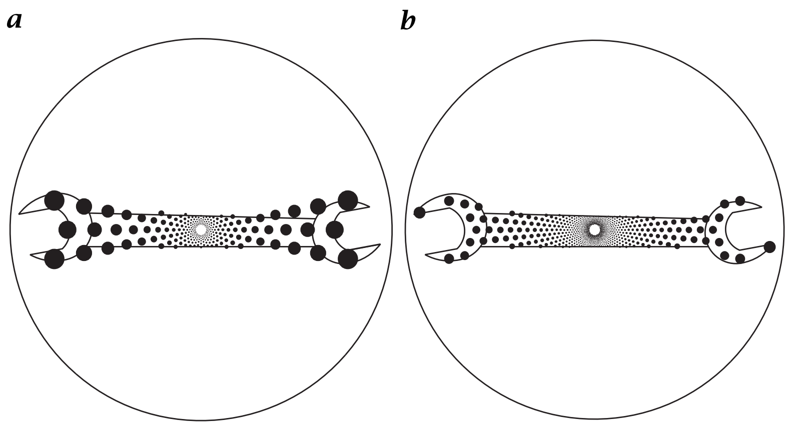


Figure 8.2: ***a*** *Under-representation of peripheral parts;* ***b*** *attempt to increase peripheral resolution simultaneously by increasing central over-representation.*

We start with an image $\boldsymbol{I}(r, s, \alpha)$ of an object of size s at distance r and in orientation α. Obviously, the radius of a disk that will be covered by $\boldsymbol{I}(r, s, \alpha)$ depends on r and s and shall be R_0 for an object of standard size s_0 at standard distance r_0. Then (with a pinhole camera, and with $r \gg s$) the radius for an arbitrary object will be

$$R_{\text{obj}}(\boldsymbol{I}) = R_0 \frac{s}{s_0} \frac{r_0}{r} = R_0 \frac{\tilde{s}}{\tilde{r}} \quad \text{with} \quad \tilde{s} = \frac{s}{s_0}, \tilde{r} = \frac{r}{r_0} \tag{8.1}$$

which is $R_{\text{obj}}(\boldsymbol{I}) = R_0$ for standard size s_0 and standard distance r_0. If this image $\boldsymbol{I}(r, s, \alpha)$ is mapped onto a normalized image $\boldsymbol{N}(s)$ by

$$\mathcal{N}(r, \alpha) = \mathcal{S}(r)\mathcal{R}(\alpha) : \boldsymbol{I}(r, s, \alpha) \to \boldsymbol{N}(s) \tag{8.2}$$

the radius of the disk $R_{\text{obj}}(\boldsymbol{N}) = R_0\tilde{s}$ is independent of r and α while it still depends on the object size s. Especially, if $\boldsymbol{I}(r, s, \alpha)$ is represented on a log-polar grid, the mapping in Eq. (8.2) onto $\boldsymbol{N}(s)$ is achieved by radial and tangential shift. To avoid the forementioned disadvantages of simple log-polar retinae, we replace $\boldsymbol{I}$ by a set of images $\boldsymbol{I} = \{\boldsymbol{I}_1, \boldsymbol{I}_2, \boldsymbol{I}_3, \boldsymbol{I}_4\}$ with resolution decreasing in steps of two from $\boldsymbol{I}_1$ to $\boldsymbol{I}_4$. We select ring-shaped regions $\boldsymbol{I}_\nu^* \subset \boldsymbol{I}_\nu$ from $\boldsymbol{I} = \{\boldsymbol{I}_1, \boldsymbol{I}_2, \boldsymbol{I}_3, \boldsymbol{I}_4\}$ in a way such that the outer and inner radii of rings I_ν^* are $R_{\text{obj}}(\boldsymbol{I})/2^{\nu-1}$ and $R_{obj}(\boldsymbol{I})/2^\nu$, respectively.

The peripheral resolution of each ring is lower by a factor of two as the resolution at the inner edge but all the rings have identical peripheral resolution and also identical inner resolution. As a result, the image $\boldsymbol{I}^* = \boldsymbol{I}_1^* \cup \boldsymbol{I}_2^* \cup \boldsymbol{I}_3^* \cup \boldsymbol{I}_4^*$ shows an approximately constant resolution, and there is no longer an over-representation of the center and an under-representation of the periphery (Fig. 8.3).

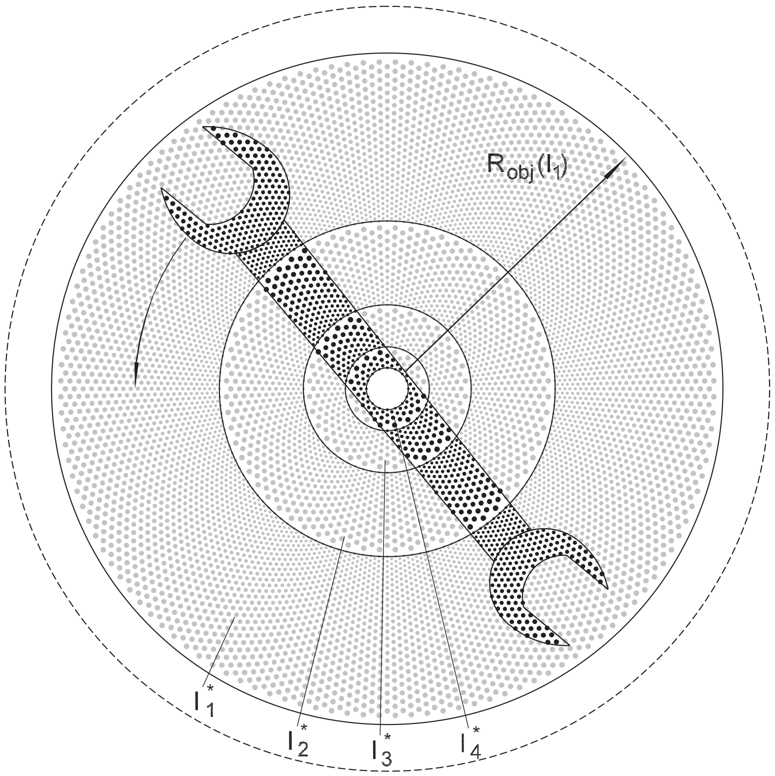


Figure 8.3: *A log-polar retina composed of rings from different spatial frequency channels shows approximately homogeneous resolution. The advantage of normalization by radial and tangential shift is still preserved.*

Of course, also the mapping in Eq. (8.2) becomes a set of mappings

$$\mathcal{S}(r)\mathcal{R}(\alpha) : \boldsymbol{I}_\nu^*(r,s,\alpha) \to \boldsymbol{N}_\nu^*(s) \quad \text{with} \quad \nu \in \{1,2,3,4\} \tag{8.3}$$

onto rings $\boldsymbol{N}_\nu^* \subset \boldsymbol{N}_\nu$ in a set $\boldsymbol{N} = \{\boldsymbol{N}_1, \boldsymbol{N}_2, \boldsymbol{N}_3, \boldsymbol{N}_4\}$ of log-polar output grids. Of course, also in $\boldsymbol{N}$ the resolution decreases in steps of two from $\boldsymbol{N}_1$ to $\boldsymbol{N}_4$ and consequently also $\boldsymbol{N}^* = \boldsymbol{N}_1^* \cup \boldsymbol{N}_2^* \cup \boldsymbol{N}_3^* \cup \boldsymbol{N}_4^*$ shows approximately constant resolution. As all the rings are appropriately scaled and rotated in Eq. (8.3), $\boldsymbol{N}^*(s)$ only depends on the object size s but not on distance r and orientation α.

Please remember that radial shift of rings does not change the number of pixels. As soon as we foveate an object (bring it to the center of our retina), we can shrink or widen our region of interest $R_{\text{obj}}(\boldsymbol{I})$ by adapting the outer radius of $\boldsymbol{I}_1^*$ to $R_{\text{obj}}(\boldsymbol{I})$ without changing the number of pixels in $\boldsymbol{I}^*$ or $\boldsymbol{N}^*$. Whatever the region of interest may be, either an object or a small detail of an object, it will always be represented by a

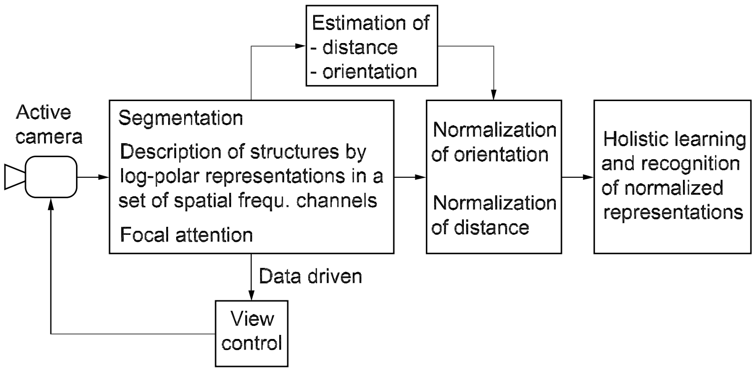

Estimation of
- distance
- orientation
Active camera
Segmentation
Description of structures by log-polar representations in a set of spatial frequ. channels
Focal attention
Normalization of orientation
Normalization of distance
Holistic learning and recognition of normalized representations
Data driven
View control

Figure 8.4: *Learning and recognition of normalized representations.*

constant number of pixels. In other words, the absolute resolution of a detail increases as soon as we shrink our region of interest to it, that is, if we "zoom out."

Now we are ready to extend the block diagram of our system by a normalization component (Fig. 8.4). Different from biology, there was no "hardware" log-polar retina available when we implemented this type of mapping.

Therefore, we extracted log-polar grids from a 512^2 image and due to limited resolution the inner radii could not be less than one-eighth of the outer radii in all the grids $\boldsymbol{I} = \{\boldsymbol{I}_1, \boldsymbol{I}_2, \boldsymbol{I}_3, \boldsymbol{I}_4\}$. By these restrictions the "zoom factor" and consequently the range of distance and object size is very limited. The total number of pixels in $\boldsymbol{I}^*$ or $\boldsymbol{N}^*$ is 8160. A detailed description of the implementation is given in Hartmann et al. [2] and a modified approach avoiding the forementioned limitations is described in Section 8.3.3.

8.2.3 The tolerance-separability problem

In the preceding chapters we have discussed normalized representations of objects and the normalization mechanisms itself. However, we have not yet mentioned by what kind of feature vector or by what kind of neural activity pattern our object is represented. In this chapter we introduce two types of feature vectors, the region-based representations and the contour-based representations. We will see that region-based representations show high tolerance against foveation and normalization errors but limited separation of similarly shaped objects. On the other hand, we will see that contour-based representations show high distinctiveness but low tolerance against the forementioned errors. Finally, we will introduce a solution for this problem that is well known from cortical representations, and for this reason we will treat

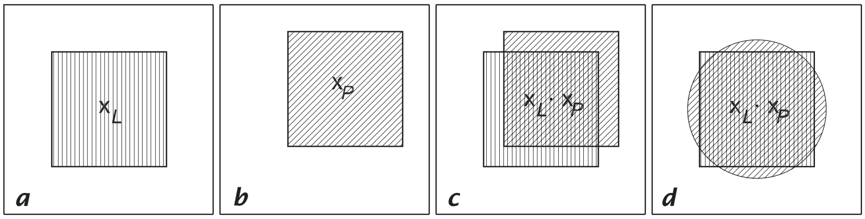


Figure 8.5: *Region-based representations are tolerant against errors in foveation and normalization (**c**). However, with growing tolerance separability decreases (**d**).*

contour-based feature vectors as activity patterns of model neurons in this chapter.

Under the constraints of this part of the chapter limited to holistic learning and recognition of segmented 2-D objects, region-based representations are very simple. We assign consecutive numbers n to all pixels of the normalized image $\boldsymbol{N}^*$. This labeling depends neither on the number of the ring in $\boldsymbol{N}^*$ nor on the special hexagonal log-polar grid. In the segmented image we set pixels $x_n = 1$ if they belong to a selected segment, and $x_n = 0$ otherwise. Consequently, the feature vector $\boldsymbol{x}$ of a region-based representation is

$$\boldsymbol{x} = [x_1, x_2, \dots, x_n, \dots, x_N] \quad \text{with} \quad x_n = \begin{cases} 1 & \text{if } x_n \in \text{segment} \\ 0 & \text{otherwise} \end{cases} \tag{8.4}$$

Of course, we will introduce more powerful region-based feature vectors in Section 8.3.1 representing gray scale and color. However, the tolerance-separability problem and its solution can be explained at the level of these simple feature vectors.

It is well known and easy to see in Fig. 8.5 that region-based representations are tolerant against inevitable errors in foveation and normalization. Assume that we have learned the vector $\boldsymbol{x}_L$ of a square-shaped object (Fig. 8.5a) and we try to recognize it in a slightly shifted position (Fig. 8.5b). Only few components in the feature vector $\boldsymbol{x}_P$ of the presented object are changed and the similarity measure $d(\boldsymbol{x}_L, \boldsymbol{x}_P)$ will not be significantly reduced. If $d(\boldsymbol{x}_L, \boldsymbol{x}_P) = \boldsymbol{x}_L \boldsymbol{x}_P$ is, for example, a simple dot product, it is characterized by the overlapping area in Fig. 8.5c. If we introduce a threshold d_0 of, say, 90% of the maximum match, then, $d(\boldsymbol{x}_L, \boldsymbol{x}_P) > d_0$ will also hold if the presented object is slightly displaced or not exactly normalized with respect to the learned object. Actually, our distance measure is “dot product – Hamming distance” and we will return to this in Section 8.2.5.

If we adapt d_0 to the expected errors we can still separate small objects from large ones, compact objects from elongated ones, etc. How-

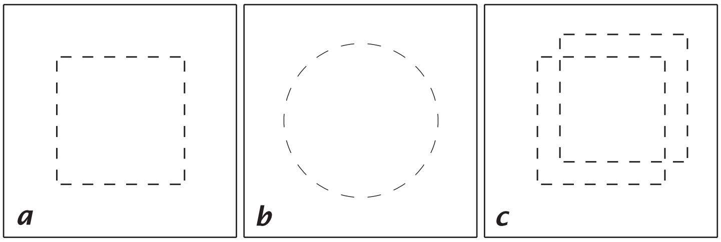


Figure 8.6: *Contour-based representations provide high separability (**a, b**) but there is not even a minimum of tolerance at all (**c**).*

ever, if we adapt d_0 to high tolerance even distinctly different objects of similar size and compactness will no longer be separated. The learned square-shaped pattern will also match sufficiently with a similarly sized circular pattern (Fig. 8.5d) and thus they will be confused. This problem can not be solved by increasing d_0 to a higher level. Of course, with a higher value of d_0 the circular- and the square-shaped area could be separated; however, tolerance would be simultaneously reduced.

The only way out of this dilemma is to introduce additional representations of objects, and a very obvious idea is to introduce contour representations. We assign a set of contour detectors to each pixel n with $l = 1 \dots L$ orientations, and we introduce a vector

$$\boldsymbol{k}_n = [k_{n1}, k_{n2}, \dots, k_{nl}, \dots, k_{nL}] \quad \text{with} \quad l \in \{1, 2, \dots, L\} \tag{8.5}$$

with components k_{nl} for each pixel. In this binary vector, $k_{nl} = 1$ describes the response of a contour detector at pixel n with orientation l or the presence of an oriented contour element, and nonresponding detectors are described by components $k_{nl} = 0$ in $\boldsymbol{k}_n$. For a complete description of all the oriented contour elements in a whole image $\boldsymbol{N}^*$, we combine all these vectors $\boldsymbol{k}_n$ to a vector $\boldsymbol{k}$

$$\boldsymbol{k} = [\boldsymbol{k}_1, \boldsymbol{k}_2, \dots, \boldsymbol{k}_n, \dots, \boldsymbol{k}_N] \quad \text{with} \quad n \in \{1, 2, \dots, N\} \tag{8.6}$$

This vector $\boldsymbol{k}$ with NL components is again a feature vector and as it describes the contour structures of $\boldsymbol{N}^*$ we call it a contour-based representation.

Now we assume that we have learned the contour-based feature vector $\boldsymbol{k}_L$ of our square-shaped area (Fig. 8.6a), and that we show the circular area again. In this case there will be no confusion as the feature vector $\boldsymbol{k}_P$ of the presented circular area is significantly different from the learned vector $\boldsymbol{k}_L$ of the square-shaped area. While the feature vector of the square-shaped area only contains elements representing horizontal and vertical orientations, the feature vector of the circular area contains components with all orientations l. Thus objects confused by region-based representations are well separated by contour-based representations. However, there is no longer the tolerance against foveation and normalization errors. If we present a slightly shifted square-shaped area, which was easily recognized in a region-based representation, there will be no match at all between the contour-based feature vectors (Fig. 8.6c). Although there are components k_{nl} representing horizontally and vertically oriented contour elements in both the vectors $\boldsymbol{k}_L$ and $\boldsymbol{k}_P$, they occur at different pixels and differ in n. The similarity measure $d(\boldsymbol{k}_L, \boldsymbol{k}_P)$ is zero and unlike in the case of region-based representations tolerant recognition can not be achieved by reducing the recognition threshold d_0. However, we have found a very advantageous solution by introducing so-called tolerant contour representations.

8.2.4 Tolerant contour representations

The idea of tolerant contour representations is based on contour representations in biological systems. Cortical neurons with oriented receptive fields [3] may be seen as detectors k_{nl} responding to contour elements of orientation l at position n in the visual field. There are sets $\boldsymbol{k}_n = [k_{n1}, k_{n2}, \dots k_{nl}, \dots k_{nL}]$ of neurons at positions n, and we can combine these sets to a set of all neurons $\boldsymbol{k} = [\boldsymbol{k}_1, \boldsymbol{k}_2, \dots \boldsymbol{k}_n, \dots \boldsymbol{k}_N]$. If there is a contour structure in the visual field, some neurons k_{nl} of set $\boldsymbol{k}$ will respond while other neurons will remain inactive. Of course, cortical neurons must not be seen as binary elements, and we will consider this in Section 8.3.3. The essential features of tolerant representations, however, can be easily explained with simplified binary neurons in this section. In this case the activity pattern can be described by a binary vector $\boldsymbol{k}$ with some components k_{nl} being 1, others being 0. Thus, our feature vector $\boldsymbol{k}$ introduced in Section 8.2.3 can also be used for the description of a neural activity pattern.

So-called simple cortical neurons or simple cells k_{nl} respond to an oriented contour element only under the condition that this stimulus shows orientation l and that it is located at the position n of the visual field. This position and orientation in the visual field is described by a rectangular window W^s_{nl}, the so-called receptive field of the simple neuron k_{nl} (Fig. 8.7a). If the contour element is running within this window W^s_{nl}, and if its orientation does not differ more than $\pm 15°$ from the longitudinal axis of the rectangle, the corresponding neuron k_{nl} will respond. Another essential feature of cortical neural representations is commonly known retinotopic mapping. This means that neighboring neurons in the cortical surface have neighboring receptive fields in the visual space. For this reason, a line in the visual field will be represented by a string-shaped activity pattern of simple cells at the cortical surface (Fig. 8.7a). It should be remembered that this representation is not at

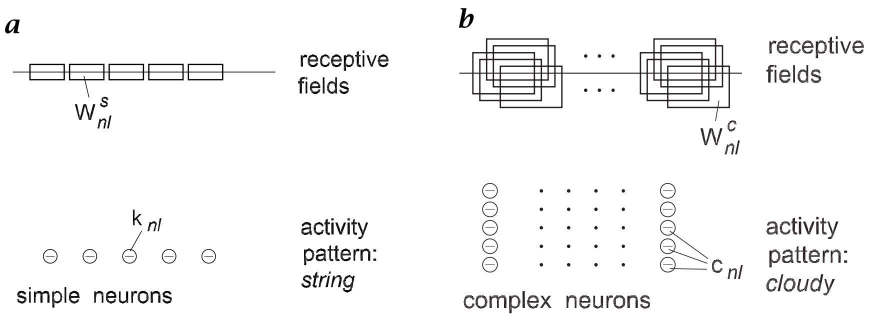


Figure 8.7: ***a*** *Simple cortical neurons respond with a string-shaped activity pattern; while* ***b*** *complex cortical neurons respond with "cloudy" activity patterns.*

all tolerant against foveation and normalization errors as we have seen in the example of Fig. 8.6.

Additionally, there are also cortical representations by thus designated complex neurons or complex cells. Again, a complex cell c_{nl} responds to a contour element if it shows correct orientation l and if it is correctly located at position n in the visual field. Again, correct position and orientation may be described by the receptive field W^c_{nl} of the complex cell (Fig. 8.7b), and, again, the activity pattern may be described by a vector

$$\begin{aligned} \boldsymbol{c} &= [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n, \ldots, \boldsymbol{c}_N], n \in \{1, 2, \ldots, N\} \quad \text{with} \\ \boldsymbol{c}_n &= [c_{n1}, c_{n2}, \ldots, c_{nl}, \ldots, c_{nL}], \quad l \in \{1, 2, \ldots, L\} \end{aligned} \tag{8.7}$$

The most essential difference between simple and complex cells is the size of the receptive fields. While the receptive fields W^s_{nl} of simple cells are small and show almost no overlap, the receptive fields W^c_{nl} of complex cells are large and heavily overlapping. The same element of a contour is covered by a whole set of receptive fields of complex cells (Fig. 8.7b), and, consequently, a "cloudy" activity pattern appears at the cortical surface.

Due to this cloudy activity pattern, however, representations by complex cells are tolerant against foveation and normalization errors. If we learn the complex contour representation $\boldsymbol{c}_L$ of a square-shaped object and present the object in a slightly shifted position (Fig. 8.8), the representation $\boldsymbol{c}_P$ will still show considerable match. Therefore, we can calculate a similarity measure $d(\boldsymbol{c}_L, \boldsymbol{c}_P)$ and, as in the case of region-based representations, we can adjust a recognition threshold d_0 to an appropriate value allowing recognition if $d(\boldsymbol{c}_L, \boldsymbol{c}_P) > d_0$. Note here that matching components c_{nl} in $\boldsymbol{c}_L$ and $\boldsymbol{c}_P$ must correspond in position and orientation, and, consequently, tolerant representations are different from sheer blurring.

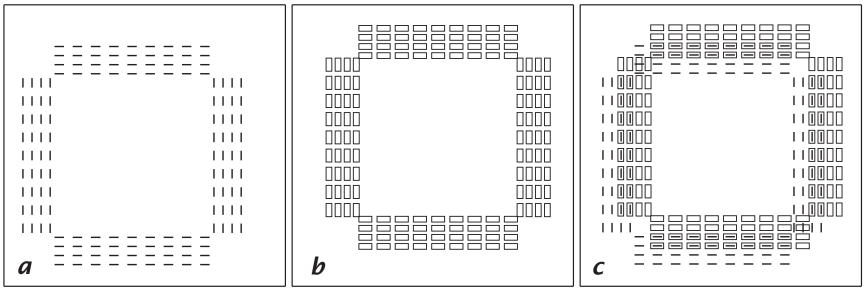


Figure 8.8: ***a*** *Tolerant contour representation of a learned; and* ***b*** *of a slightly shifted square-shaped structure.* ***c*** *Due to the "cloudy" activity pattern there is still overlap.*

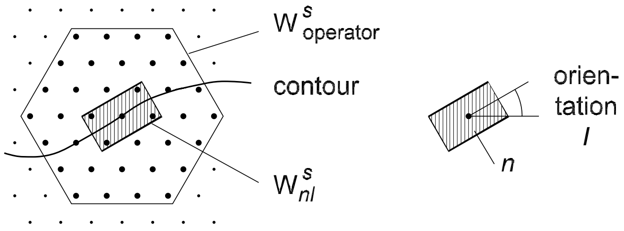


Figure 8.9: *Contour elements in the receptive field W^s_{nl} are detected by verification of the sign combinations in the operation window W^s_{op}.*

Until now, we have discussed only the features of simple and complex neurons but not their technical realization. Although we have introduced tolerant contour representations on a biological background, we will not speculate on biological implementations. What we actually are interested in are operations on our normalized image $\boldsymbol{N}^*$ providing comparable features. In the case of artificial simple cells we use oriented contour detectors based on local operations. Postponing the discussion of analog Gabor-based detectors to Section 8.3.3, we introduce a binary detector set operating on a hexagonal pixel matrix. In an operation window W^s_{op} that contains three rings of pixels around the central pixel n (Fig. 8.9), we provide the Laplacian image. Contour elements are detected by verification of the sign combinations in W^s_{op}. It is an essential feature of this detector set that arbitrarily running contours are completely enveloped by continuous chains of smoothly fitting "receptive fields" W^s_{nl} and hence arbitrary contours are completely represented. All the other features of this detector set are described in more detail by Hartmann [4, 5].

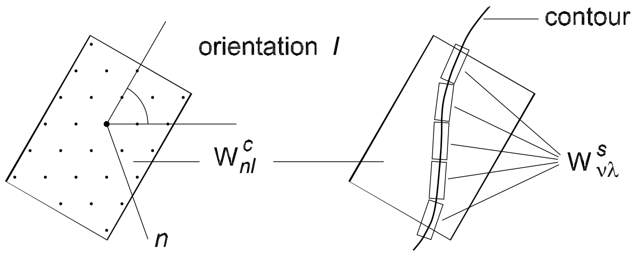


Figure 8.10: *Our artificial complex cell responds if the receptive fields of responding simple cells form a continuous chain running completely within its receptive field* W^c_{nl}.

Before we proceed to operations providing tolerant representations $\boldsymbol{c}$ of the normalized image $\boldsymbol{N}^*$, we must discuss in more detail those features of complex cells that are essential in our technical implementation. First, enlarged receptive fields W^c_{nl} of complex cells must not be confused with receptive fields of detectors responding to contours in lower spatial frequency channels. In spite of enlarged receptive fields complex cells respond to comparably fine contour structures. Second, complex cells only respond to continuous contours running through the receptive field W^c_{nl} without interruption. Finally, the orientation of the contour must not differ more than $\pm 15°$ from the longitudinal axis of the receptive field. All these features are simultaneously provided by the following operation.

We define a large rectangular receptive field W^c_{nl} with orientation l centered at pixel n (Fig. 8.10). We apply an operation $\mathcal{O}^c$ to all those simple cells $k_{\nu\lambda}$ centered at pixels ν inside of W^c_{nl}. Operation $\mathcal{O}^c$ verifies whether the receptive fields of responding simple cells $k_{\nu\lambda}$ form smooth chains running in the axial direction of W^c_{nl}. The set of all these sequences of simple receptive fields describes all possible uninterrupted fine contour structures running through W^c_{nl} within an orientation tolerance of $\pm 15°$ (Fig. 8.11). If $\mathcal{O}^c$ detects an active sequence the complex cell is active and c_{nl} is set to 1

$$c_{nl} = \mathcal{O}^c(k_{\nu\lambda}) = \begin{cases} 1 & \text{if active sequence} \in W^c_{nl} \\ 0 & \text{otherwise} \end{cases}, \quad \nu \in W^c_{nl} \tag{8.8}$$

One possible implementation of operation $\mathcal{O}^c$ is based on the forementioned detector set that allows verification of active sequences very simply. Note here that the original algorithm [5] is modified. The distance of the "rosettes" no longer increases with higher linking levels and this leads to the highly overlapping receptive fields W^c_{nl}. There are also hardware implementations as well for the detector set as for the operation $\mathcal{O}^c$.

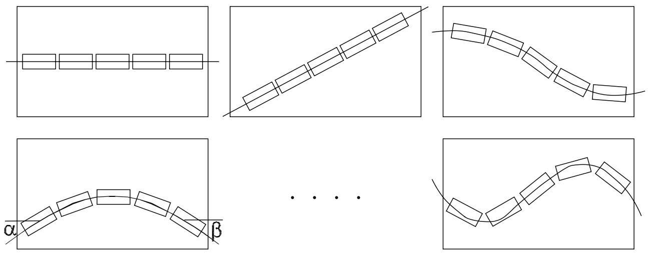


Figure 8.11: *Some examples for smooth sequences of receptive fields. Corners (vertices) can be detected by comparing the orientations α, β at both ends of the sequence.*

With a slight modification of the operation $\mathcal{O}^c$ we can provide an additional representation of vertices. Again, we look for smooth sequences of receptive fields but we also compare the orientation at both ends of the sequence in W^c_{nl}. If the difference exceeds a threshold, we represent this local region by a component v_n of a vector $\boldsymbol{v}$, if not, we represent it by c_{nl} as before

$$\begin{aligned} c_{nl} &= \mathcal{O}^c(k_{\nu\lambda}) = \begin{cases} 1 & \text{if straight sequence} \in W^c_{nl} \\ 0 & \text{otherwise} \end{cases}, \quad \nu \in W^c_{nl}, \\ v_n &= \mathcal{O}^v(k_{\nu\lambda}) = \begin{cases} 1 & \text{if curved sequence} \in \bigcup_{l=1}^{L} W^c_{nl} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{8.9}$$

The additional vertex representation is again described by a vector $\boldsymbol{v} = [v_1, \ldots v_n, \ldots v_N]$ and due to the overlapping receptive fields also the vertex representation is tolerant against foveation and normalization errors. Figure 8.12 gives an example of the “cloudy” contour representation of a Cranfield part (see also Fig. 8.29).

Another outstanding feature of tolerant contour representations are their “combinatorical filtering”. According to Eq. (8.8), elements c_{nl} are only encoded if continuous contours are running through the receptive field W^c_{nl} and there is no response if there are broken lines or textures. Consequently, tolerant representations only encode essential contour structures and suppress small details and textures. This is very useful as only the essential features of an object are learned and not the accidental variations.

This “combinatorical filtering” is also essential for orientation estimation by evaluation of orientation histograms (Fig. 8.12). If we use orientation histograms based on tolerant representations $\boldsymbol{c}$, only essential contour structures are evaluated. If we would use histograms based on representations $\boldsymbol{k}$, background and peak position would be influenced by orientation of textures or by orientation of minor details.

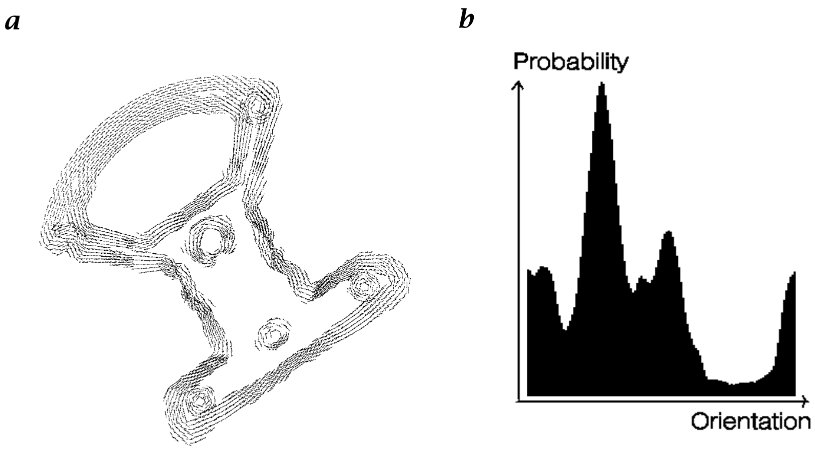


Figure 8.12: *Evaluation of peak position in an orientation histogram provides robust estimation of orientation.*

There are some suggestions that biological complex cells also may be based on linking mechanisms and that linking could be realized by synchronization of spikes [6].

8.2.5 Learning and recognition of normalized representations

In the preceding sections we have introduced region-based representations (Eq. (8.4)), which are tolerant against errors in foveation and normalization. We also find operations providing tolerant contour and vertex representations (Eq. (8.9)), and we keep in mind that all these vectors $\boldsymbol{x}$, $\boldsymbol{c}$, and $\boldsymbol{v}$ are extremely long vectors. While region-based representations are very robust in discrimination between differently sized objects or between objects of different compactness, contour and vertex representations are able to distinguish similarly sized objects such as square shaped and circular areas. As all these representations show special advantages we combine them to a representation $\boldsymbol{t} = [\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{v}]$, simply called tolerant representation.

The question is now how to learn this extremely long vector $\boldsymbol{t}$ or its components $\boldsymbol{x}$, $\boldsymbol{c}$, and $\boldsymbol{v}$. The key idea of tolerant representations is to shift generalization away from learning and to provide it by the special means of representation. This is quite different from the well-known neural network approach, where varying representations are generalized by the network during extended training sequences. In our approach variations of representations due to foveation and normalization errors, but also due to minor perspective distortions and variations in object shape, are simply tolerated by the construction of the representation itself. Accordingly, vectors $\boldsymbol{t}$ can simply be “stored” by any kind of one-shot learning, and—only to repeat—there are no training sequences at all. If there is no training of networks, however, there are no convergence problems and hence no limitation with respect to the length of feature vectors.

Obviously, we not only want to learn representations of objects, but we also want to recognize objects; thus we have to discuss in more detail the similarity measure $d(\boldsymbol{t}_L, \boldsymbol{t}_P)$ between learned representations $\boldsymbol{t}_L$ and representations $\boldsymbol{t}_P$ of momentarily presented objects. For two reasons we prefer a similarity measure depending on "dot product minus Hamming distance." First, this measure is provided by the closed-loop antagonistic network (CLAN) architecture, and, second, it is intuitively appropriate. CLAN is an unsupervised classification network [7] that responds dynamically to changing feature vectors by either assigning them to existing classes or by automatically creating new classes. The dynamics for selection of the winner are determined by a closed-loop architecture, and an antagonistic representation provides the forementioned similarity measure. The advantage is seen intuitively in the case of region-based representations. Let the number of active components in the presented representation be $p = |\boldsymbol{x}_P|$, the number of active components in the learned representation be $l = |\boldsymbol{x}_L|$, and the number of matching active components be m. Now our similarity measure

$$\begin{aligned} d(\boldsymbol{x}_L, \boldsymbol{x}_P) &= m - [(p - m) + (l - m)]/p \\ &= [\boldsymbol{x}_L \boldsymbol{x}_P - (\boldsymbol{x}_L - \boldsymbol{x}_P)^2] \,/\, |\boldsymbol{x}_P| \end{aligned} \tag{8.10}$$

is easily explained by a Venn diagram (Fig. 8.13). Match m between pixels of the learned segment and the presented segment is rewarded, mismatch of $(p - m)$ and $(l - m)$ is punished. As the expression $m - [(p - m) + (l - m)]$ grows with the size of the segment it is divided by the number of pixels p of the presented segment. Note that the similarity measure decreases with increasing mismatch even if the match remains constant (Fig. 8.13), which is in good agreement with intuition. Of course, the same arguments are valid if similarly we calculate similarity measures $d(\boldsymbol{c}_L, \boldsymbol{c}_P)$ and $d(\boldsymbol{v}_L, \boldsymbol{v}_P)$ between tolerant contour and vertex representations. We also can calculate $d(\boldsymbol{t}_L, \boldsymbol{t}_P)$ between learned and presented combined representations $\boldsymbol{t} = [\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{v}]$, but sometimes it is more convenient to calculate the individual measures and to combine the results by weighted summation.

The forementioned measure is provided by the CLAN, which we used when we began investigations with our recognition system. We could achieve a considerable speed-up, however, by simply storing the feature vectors in the learning phase, and by calculating the similarity measure between stored and presented vectors. To describe our system for holistic learning and recognition we again have to extend our block diagram (Fig. 8.14).

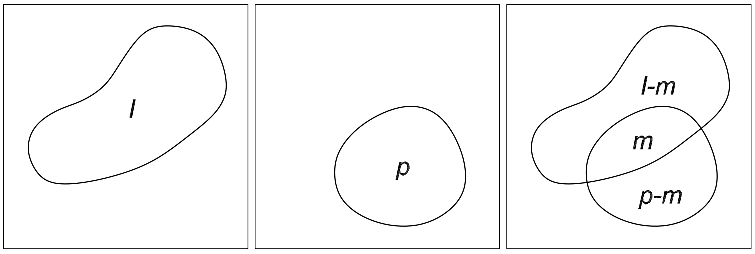


Figure 8.13: *The similarity measure of the CLAN rewards match and punishes mismatch between learned and presented representations.*

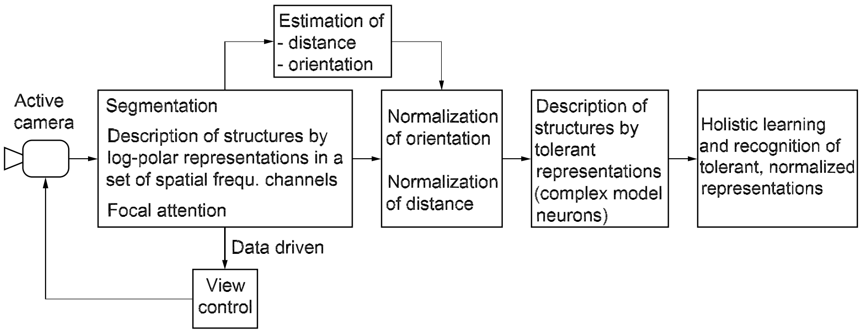

Estimation of
- distance
- orientation
Active camera
Segmentation
Description of structures by log-polar representations in a set of spatial frequ. channels
Focal attention
Normalization of orientation
Normalization of distance
Description of structures by tolerant representations (complex model neurons)
Holistic learning and recognition of tolerant, normalized representations
Data driven
View control

Figure 8.14: *Normalized tolerant representations are holistically learned.*

There is at least some similarity between tolerant contour representations in our system and contour representation by complex cells in the visual system. If there should also be similarities between our architecture and architectures of biological visual systems, we should mention that the blocks "normalization" and "tolerant representation" (Fig. 8.14) seem to be interchanged in biological systems.

8.3 Holistic recognition of 3-D objects in real scenes

8.3.1 Introduction of color

The basic system as it is described in Section 8.2 shows good performance in industrial scenes where well-segmented parts are handled under good illumination conditions and before an unstructured background. Under these conditions region representations based on binary segmentation are sufficient. Introduction of color, however, improves segmentation as well as discrimination between similarly shaped objects.

There is no specifically developed color segmentation algorithm in our system and so only our strategy of color segmentation is sketched. We assume that nonoverlapping objects are segmented from a uniformly colored background. Pixels are color labeled in the hue-intensity-saturation color space (HIS space) and the color of the background is evaluated by histogram techniques. The background is segmented based on the color labeling of the pixels, and the remaining areas not belonging to the background are assigned to objects. By this simple strategy also multicolored objects are described by compact segments and are not divided into differently colored parts. This is essential for holistic learning of multicolored objects as we will see in the following. This simple segmentation in the color space is more reliable and robust compared with segmentation in gray-scale images. The restriction of this strategy to nonoverlapping parts in front of a uniform background can be tolerated in many industrial applications, and we will eliminate this restriction totally in Section 8.4.

As already mentioned, we also use color to improve discrimination between similarly shaped objects. More exactly, in $\boldsymbol{t} = [\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{v}]$ we exchange our binary region-based representation $\boldsymbol{x}$ for a tolerant color representation $\boldsymbol{p}$. It proved advantageous to use a color representation based on the HIS space. This allows a color classification that is more or less independent of the absolute intensity of illumination or of shading. On the other hand, hue values are not defined for the black-white axis, and become very uncertain if they are close to it. Therefore, we represent a pixel n by its hue h_n, as long as hue is well defined, and by its intensity otherwise. Hue is considered to be well defined if the intensity i_n as well as the saturation s_n exceed corresponding thresholds i_0 and s_0. Statistical evaluations show that approximately 95 % of all pixels are encoded by hue and only 5 % by intensity. However, both hue and intensity are analog-valued variables incompatible with our binary feature vectors. Thus we decided to introduce a binary color representation by assigning h_n and i_n to intervals represented by components of a binary vector $\boldsymbol{p}_n$.

More exactly, we fix 25 equidistant hue values $H_0, H_1, \ldots H_{24}$ defining limits of 24 hue intervals and assign h_n to one of these intervals h_{nj}. If hue is assigned to interval h_{nj}, the jth component p_{nj} of the binary feature vector is set to 1. This very simple feature vector, however, shows a crucial disadvantage. If pixel n is represented by $p_{nj} = 1$ in the learned vector $\boldsymbol{p}_L$ and by $p_{nj-1} = 1$ or by $p_{nj+1} = 1$ in the presented vector $\boldsymbol{p}_P$ due to variations in illumination, $\boldsymbol{p}_L$ and $\boldsymbol{p}_P$ will no longer match at this position and there is no "color tolerance" in this simple representation. In order to make the color representation tolerant we not only set $p_{nj} = 1$ if h_n belongs to the jth interval, but we also set the neighboring components $p_{nj-1} = p_{nj+1} = 1$ in the feature vector $\boldsymbol{p}$

$$p_{nj} = \begin{cases} 1 & \text{if} \quad H_{j-1} \le h_n < H_j \quad \text{for} \quad j = 1, 2, \ldots 24 \\ & \text{and if} \quad (i_n > i_0) \wedge (s_n > s_0) \\ 0 & \text{otherwise} \end{cases} \tag{8.11}$$

$$\begin{aligned} p_{nj-1} &= 1 \quad \text{if} \quad p_{nj} = 1 \quad \text{with} \quad (j-1) = 24 \quad \text{if } j = 1 \\ p_{nj+1} &= 1 \quad \text{if} \quad p_{nj} = 1 \quad \text{with} \quad (j+1) = 1 \quad \text{if } j = 24 \end{aligned}$$

If either saturation s_n or intensity i_n are lower than the threshold i_0 or s_0 intensity replaces hue, intensity is encoded by the components 25 to 32 of $\boldsymbol{p}_n$ in the same way that hue is encoded by components 1 to 24. Again, there are equidistant intensity values $I_0, I_1, \ldots I_6$ defining limits of six intensity intervals and assigning i_n to one of these intervals. If intensity is assigned to the jth interval, the $(j+25)$-th component p_{nj+25} of the binary feature vector is set to 1. Again, neighboring components are set to one also in order to provide tolerance against minor intensity variations

$$p_{nj+25} = \begin{cases} 1 & \text{if} \quad I_{j-1} \le i_n < I_j \quad \text{for} \quad j = 1, 2, \ldots 6 \\ & \text{and if} \quad (i_n \le i_0) \vee (s_n \le s_0) \\ 0 & \text{otherwise} \end{cases} \tag{8.12}$$

$$\begin{aligned} p_{nj+25-1} &= 1 \quad \text{if} \quad p_{nj+25} = 1 \\ p_{nj+25+1} &= 1 \quad \text{if} \quad p_{nj+25} = 1 \end{aligned}$$

The tolerant color representation is a vector $\boldsymbol{p} = [\boldsymbol{p}_1, \ldots \boldsymbol{p}_n, \ldots \boldsymbol{p}_N]$ with the earlier-defined components $\boldsymbol{p}_n = \left[p_{n1}, \ldots p_{nj}, \ldots p_{n32}\right]$ in which components are either set by color according to Eq. (8.11) or by intensity according to Eq. (8.12). This vector $\boldsymbol{p}$ replaces $\boldsymbol{x}$ in $\boldsymbol{t} = [\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{v}]$ and we get a tolerant representation $\boldsymbol{t} = [\boldsymbol{p}, \boldsymbol{c}, \boldsymbol{v}]$. With this color representation also multicolored segmented objects can be learned and recognized holistically and this representation is tolerant also with respect to minor variations in hue and intensity.

8.3.2 View-based recognition of three-dimensional objects

In Section 8.2.4 it was already mentioned that our representations $\boldsymbol{t}$ not only tolerate foveation and normalization errors but also minor perspective distortions. This is very convenient for the extension of our system to 3-D applications by a view-based approach. View-based strategies assume that 3-D objects are represented by a set of learned views, and that additional views from other directions may be recognized by interpolation between neighboring learned views. In our system the set of learned views is a set of tolerant representations $\boldsymbol{t}_L$ that

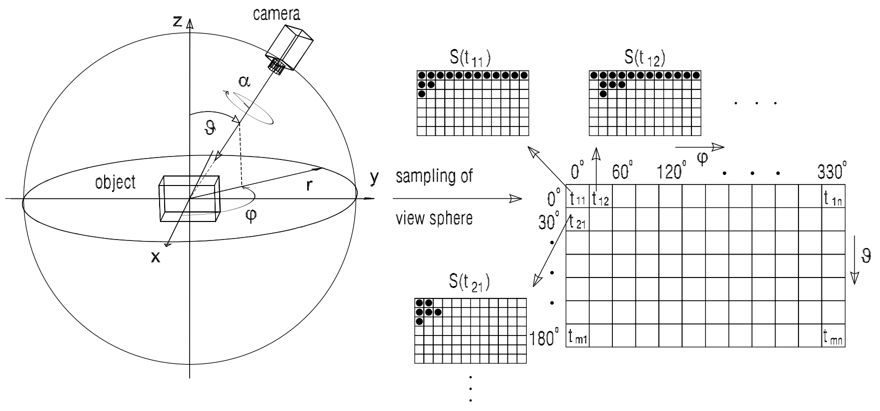


Figure 8.15: Views are learned from different positions (ϑ, φ) of the view sphere. Different learned views cover differently sized and shaped regions of the view sphere.

are also reliably recognized from slightly different directions. Therefore we need no special module for interpolation between neighboring views, and for this reason our system is excellently qualified without any change for view-based 3-D recognition.

We actually have not had to make any extension of our system. We only have to find a strategy for covering the view sphere with a minimum number of learned views. Generally speaking, the relative position of the camera with respect to the object can be changed in six degrees of freedom. By foveation of the object we already remove two degrees of freedom and we can describe the situation by four variables $(r, \alpha, \vartheta, \varphi)$ as shown in Fig. 8.15. We need not learn different views from different distances r due to distance invariance of our system. Finally, orientation invariance makes recognition independent of rotation α of the camera around its axis. We only have to learn views from different positions (ϑ, φ) on the sphere.

The strategies to find a minimum number of views covering the sphere is given in more detail by Dunker et al. [8], thus here we only show the essential steps. First, the view sphere is tessellated and for simplicity ϑ and φ are divided into equally sized intervals (Fig. 8.15). Then we learn all the views $\boldsymbol{t}_{11}, \boldsymbol{t}_{12}, \ldots \boldsymbol{t}_{1n}, \ldots \boldsymbol{t}_{mn}$ and evaluate the corresponding "covering sets" $S(\boldsymbol{t}_{11}), S(\boldsymbol{t}_{12}), \ldots S(\boldsymbol{t}_{1n}), \ldots S(\boldsymbol{t}_{mn})$. A covering set $S(\boldsymbol{t}_{mn})$ is a set of all those elements of the tessellation (black in Fig. 8.15, right-hand side) from which the object is recognized by the learned representation $\boldsymbol{t}_{mn}$. As one view $\boldsymbol{t}_{mn}$ covers more than one element of the view sphere we obviously do not need to store all the learned views. Then the question is what views should be eliminated in

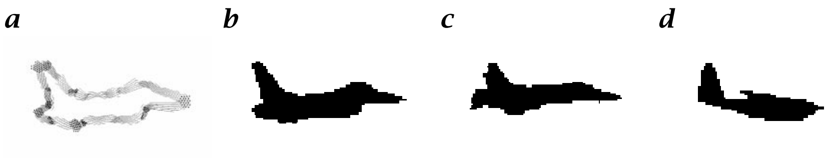


Figure 8.16: ***a*** *Tolerant representation of an airplane silhouette;* ***b–d*** *examples of different types of planes.*

order to keep a minimal complete set of prototypical views? Very good solutions for this NP-complete problem could be found by genetic algorithms [8] but results of similar quality could be achieved by a simple heuristic: we select that set $S(\boldsymbol{t}_{mn})$ covering the maximum number of elements of the tessellation and map all these positions onto a previously empty covering base. Then we search for that covering set that covers a maximum of additional elements in the covering base. We continue this procedure until all elements in the covering base are labeled. This shows that the previously selected set of covering sets covers the view sphere, and that the corresponding subset of views is sufficient for recognizing the object from arbitrary positions. The size of this subset depends on the complexity and on the symmetry of the object. Obviously, we need only one prototypical view for a ball, and typically the number is between 20 and 60.

The approach of learning prototypical sets of tolerant views has proved to be very powerful. We can demonstrate this with a set of airplane silhouettes made available to us by M. Seibert, MIT Lincoln Laboratory. This set contains more than 3600 views of F-16, F-18 and HK-1 planes (Fig. 8.16) from all parts of the view sphere. From a randomly selected training set of about 1000 views we generated a set of prototypical views for each plane. In order to distinguish between the very similar F-16 and F-18 we had to provide a relatively high number of 58 and 67 prototypes. The significantly different HK-1 was sufficiently represented by 32 prototypical views. With the remaining test set of more than 2000 silhouettes we could achieve recognition rates of 93.3 % (F-16), 95.4 % (F-18), and 99.4 % (HK-1).

8.3.3 Gabor-based contour representations

In Section 8.3.1, we have replaced the binary region-based representation $\boldsymbol{x}$ by a color representation $\boldsymbol{p}$ in order to improve our tolerant representation $\boldsymbol{t}$. We can now replace the contour representations $\boldsymbol{k}$ and $\boldsymbol{c}$ by Gabor-based representations (Volume 2, Section 4.2.2) that proved to be advantageous in real-world scenes with low contrast and blurred edges. Gabor filters, however, are not compatible with log-polar pixel arrays. Therefore we have to change our normalization strategy.

We represent our image $\boldsymbol{I} = \boldsymbol{I}(r, s, \alpha)$ on an orthogonal pixel matrix and, consequently, rotation $\mathcal{R}(\alpha)$ and scaling $\mathcal{S}(r)$ can be well-known image processing operations. They provide a normalized orthogonal image $\boldsymbol{N}(s)$ according to $\mathcal{N}(r, \alpha) = \mathcal{S}(r)\mathcal{R}(\alpha) : \boldsymbol{I}(r, s, \alpha) \to \boldsymbol{N}(s)$. Moreover, we will see in Section 8.3.4 that we can also recognize objects without normalization of orientation $\mathcal{R}(\alpha)$ and we will replace distance normalization $\mathcal{S}(r)$ by variation of camera zoom z or by changing the camera distance r. However, there is still the problem of limited resolution and this makes normalization to $\boldsymbol{N}(s)$ impossible, as small-sized objects would be projected to a small region R_{obj} with low relative resolution while large-sized objects would cover the whole matrix $\boldsymbol{N}(s)$.

Thus, we have changed our normalization strategy and adapted $R_{\text{obj}}(\boldsymbol{I}) = R_0 z\tilde{s}/\tilde{r}$ to a fixed region R_{fix} covering $\boldsymbol{N}$. Note that in contrast to Section 8.2.2, the normalized image is now independent of object size s, and that $\boldsymbol{I}(r, s, \alpha, z)$ and $\boldsymbol{N}$ are the same pixel matrix before and after changing zoom z or distance r. Similarly, $r(\boldsymbol{I})$ and $z(\boldsymbol{I})$ are distance and zoom before scaling while $r(\boldsymbol{N})$ and $z(\boldsymbol{N})$ are the corresponding values after scaling. The scale factor $R_{fix}(\boldsymbol{N})/R_{obj}(\boldsymbol{I})$ is given by an estimation of $R_{obj}(\boldsymbol{I})$. As $R_{fix}(\boldsymbol{N}) = R_0(r_0/\tilde{s})(z(\boldsymbol{N})/r(\boldsymbol{N})$ and $R_{\text{obj}}(\boldsymbol{I}) = (R_0 r_0/\tilde{s})(z(\boldsymbol{I})/r(\boldsymbol{I}))$, a condition for zoom $z(\boldsymbol{N})$ and distance $r(\boldsymbol{N})$ is given by

$$\frac{z(\boldsymbol{N})}{r(\boldsymbol{N})} = \frac{z(\boldsymbol{I})}{r(\boldsymbol{I})} \frac{R_{\text{fix}}(\boldsymbol{N})}{R_{\text{obj}}(\boldsymbol{I})} \tag{8.13}$$

This modified normalization

$$\mathcal{S}\left(r(\boldsymbol{I}), z(\boldsymbol{I}), s\right) \mathcal{R}(\alpha) : \boldsymbol{I}(r, s, \alpha) \to \boldsymbol{N} \tag{8.14}$$

allows high variability of object size s and distance r but $\boldsymbol{N}$ and hence the learned feature vector no longer depend on object size s. However, from $R_{\text{fix}}(N)$ the object size

$$s = \frac{r(\boldsymbol{N})}{z(\boldsymbol{N})} R_{\text{fix}}(\boldsymbol{N}) \frac{s_0}{R_0 r_0} \tag{8.15}$$

is easily evaluated. Thus we can store our feature vectors at different locations according to different intervals of object size, and during recognition we only compare vectors from the appropriate subset. This stratagem provides again distance invariance instead of size invariance and once more allows us to distinguish differently sized objects of similar shape. It should be mentioned that we learn feature vectors from images with $N = 128^2$ pixels. We provide three images simultaneously (Fig. 8.17, and selection of one of these images is equivalent to selection of zoom factors $z_1, z_2, z_3 = 1, 2, 4$.

With this modified normalization strategy we are able to use orthogonal images and more advanced Gabor-based contour representations.

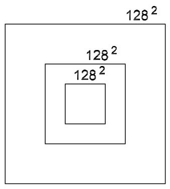


Figure 8.17: *From a 512^2 image we simultaneously calculate half-sized and quarter-sized images with 128^2 pixels.*

The image $\boldsymbol{N}$ is filtered by a fast Gabor algorithm in the Fourier domain. At each pixel n there are 12 differently oriented filters with orientation l, providing phase and amplitude values. The phase values are used for stereo-based distance measurement and not considered for object recognition. The amplitudes g_{nl} at pixel n form a vector $\boldsymbol{g}_n = [g_{n1}, \ldots g_{nl}, \ldots g_{n12}]$ and all these vectors $\boldsymbol{g}_n$ are combined to a vector $\boldsymbol{g} = [\boldsymbol{g}_1, \ldots \boldsymbol{g}_n, \ldots \boldsymbol{g}_N]$. As the Gabor filters are highly overlapping a contour element is represented also by responses of filters located in the vicinity of the contour. This makes $\boldsymbol{g}$ similar to the complex contour representation $\boldsymbol{c}$ that was introduced in Section 8.2.4. On the other hand, $\boldsymbol{g}$ is an analog-valued vector while $\boldsymbol{c}$ is binary. In order to be compatible with the previously introduced binary feature vectors, we simply replace the definition of $\boldsymbol{c}$ in Eq. (8.8) by

$$c_{nl} = \begin{cases} 1 & \text{if } g_{nl} \geq t \\ 0 & \text{otherwise} \end{cases} \tag{8.16}$$

where t is either a fixed or an adaptive threshold. This very simple tolerant representation $\boldsymbol{c}$ proved to be superior to the old one in the case of low contrast images with blurred edges. On the other hand, the advantage of "combinatorical filtering" is lost as there is no verification of continuous chains of contour elements selecting "essential contours." Therefore, we are currently investigating more sophisticated Gabor-based tolerant representations.

Obviously, we can simply replace the tolerant representations according to Eq. (8.8) by those according to Eq. (8.16) without any other change of the system. It proved to be advantageous, however, to learn "simple representations" $\boldsymbol{k}$ and to compare them with complex representations $\boldsymbol{c}$ in the recognition phase. This is easily explained by a simple experiment. Assume we have learned a vertical line by a tolerant representation $\boldsymbol{c}_L$. Now, we shift the line from left to right over the original position and compare its representations $\boldsymbol{c}_P$ at slightly different relative positions. The overlap $(\boldsymbol{c}_L \boldsymbol{c}_P)$ will increase linearly, reach a maximum, and then decrease linearly again. If we learn the "thin"

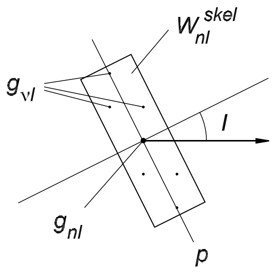


Figure 8.18: *Skeletonizing Gabor representations.*

simple representation $\boldsymbol{k}$ instead of the "cloudy" complex representation $\boldsymbol{c}$, the match $(\boldsymbol{k}_L \boldsymbol{c}_P)$ will immediately jump to a maximum value, remain constant as long as $\boldsymbol{c}_P$ overlaps $\boldsymbol{k}_L$,and jump to zero again. Evidently, the "box-shaped" tolerance profile provides better recognition of slightly shifted contours than the "roof-shaped" profile. For this reason we learn "thin" representations $\boldsymbol{k}$ and compare them with "cloudy" representations $\boldsymbol{c}$ from now on.

The only question now is how to provide "thin" representations on the base of Gabor representations $\boldsymbol{g}$. A very simple solution is the skeletonizing of $\boldsymbol{g}$ by the following algorithm. We define an elongated window W_{nl}^{skel} with a length of one Gabor period, and a width of one pixel distance perpendicular to the orientation of filter g_{nl} at pixel n (Fig. 8.18). In this window, we compare the responses of all filters with orientation l. If g_{nl} at the central pixel n exceeds the other responses, $k_{nl} = 1$ is set in the "thin" representation.

$$k_{nl} = \begin{cases} 1 & \text{if } g_{nl} > g_{\nu l} \\ 0 & \text{otherwise} \end{cases}, \quad \nu \in W_{nl}^{skel} \tag{8.17}$$

Although matching between learned "thin" representations $\boldsymbol{k}_L$ and presented "cloudy representations" $\boldsymbol{c}_P$ provides good results there remains one problem. We introduced Gabor-based representations for better performance in real-world scenes. In real-world scenes, however, there is usually no homogeneous background and correct segmentation of objects becomes difficult or even impossible. For this reason we can no longer select the center of gravity of a segment as the foveation point, and consequently shift invariance is now a major problem. Starting from the observation that our vision system does not really need a high precision foveation point, we provide a solution to this problem in the following section.

8.3.4 Extrafoveal recognition

It is well known that our vision system is able to recognize a learned object immediately even if we look almost to the edge of the object and not to its center. On the other hand, there is no recognition if we look too far away from it. If the object is recognized not exactly at the center of the visual field, that is, the so-called fovea, we refer to “extrafoveal recognition.” The first idea is obviously to increase spatial tolerance by increasing the widths of our cloudy representations but this leads immediately to the tolerance-separability problem discussed in Section 8.2.3. But the tolerance of our representations allows a solution without loss of separability.

We compare the presented representation $\boldsymbol{c}_P$ not only with one version of the learned representation but with a matrix of horizontally and vertically shifted versions $\boldsymbol{k}_L^{(\lambda\mu)}$. This method does not seem very attractive at first sight because of the waste of computing power. However, the tolerance of $\boldsymbol{c}_P$ and the box-shaped tolerance profile (see Section 8.3.3) allows a small number of rows λ and columns μ, and the comparison of binary vectors accelerates evaluation. Additionally, the procedure is accelerated by starting with images at low resolution, with 32^2 or 64^2 pixels. Comparison at low resolution is very fast and provides reliable hypotheses, which, of course, must be verified at high resolution.

The dominant contribution to acceleration, however, is achieved by a modified strategy for evaluation of the similarity measure between $\boldsymbol{c}_P$ and $\boldsymbol{k}_L^{(\lambda\mu)}$. But before discussing a modified evaluation we have to mention a modification of the similarity measure itself. In Section 8.2.5 there were three reasons for using dot product minus Hamming distance. The CLAN was available when we started our investigations and it provides this and only this measure. A measure based on match minus mismatch is very plausible, and finally we achieved excellent results with recognition of segmented objects in front of an unstructured background. In real-world scenes, which we are discussing in this section, segmentation is difficult or impossible. Thus we compare learned objects not longer with segmented objects but with complex scenes including the objects. As a result the unpredictable background structures introduce unpredictable mismatch, which also reduces our old similarity measure in cases of perfect match between a learned and a presented object. For this reason we use a similarity measure

$$d(\boldsymbol{c}_P, \boldsymbol{k}_L^{(\lambda\mu)}) = \boldsymbol{c}_P \cdot \boldsymbol{k}_L^{(\lambda\mu)} / l^{(\lambda\mu)} \quad \text{with} \quad l^{(\lambda\mu)} = |\boldsymbol{k}_L^{(\lambda\mu)}| \tag{8.18}$$

in which only the match is considered and is normalized to the number $l^{(\lambda\mu)}$ of active components in $\boldsymbol{k}_L^{(\lambda\mu)}$. Of course, this measure will only provide good results if the object is learned before a uniform background. Otherwise, the measure in Eq. (8.18) would expect the background structure of the learning situation also to be in the analyzed scene.

Now we return to the forementioned acceleration by a modified evaluation of the similarity measure. Until now, we have treated $d(\boldsymbol{c}_P, \boldsymbol{k}_L^{(\lambda\mu)})$ as a dot product between binary vectors with some hundred thousands of components. Evaluation of this dot product requires comparison of all components in pairs and also is time consuming in the case of binary vectors. However, our extremely long feature vectors show only a very small number of active components. As more than 90 % of the components of our feature vectors describe contour representations the sparse activation is easily explained. Contour structures cover only a small fraction of pixels. At those pixels with supraliminal Gabor responses g_{nl} only two or three neighboring orientations l out of twelve are represented. As a result, only a few percent of some hundred thousands of components in $\boldsymbol{c}_P$ are active. This percentage is reduced even more if we consider the learned representation $\boldsymbol{k}_L$. Due to the homogeneous background during learning the number of active components in $\boldsymbol{k}_L$ is typically smaller than in $\boldsymbol{c}_P$. Additionally, the number of components in $\boldsymbol{k}_L$ is reduced more to 20 % by skeletonizing (Eq. (8.17)). As a result, the number of active components in $\boldsymbol{k}_L$ is typically a few thousand. The correctly shifted representation $\boldsymbol{k}_L^{(\lambda\mu)}$ should have a high percentage of components $k_{nl}^{(\lambda\mu)}$ matching active partners c_{nl} in $\boldsymbol{c}_P$ while all the differently shifted representations should have fewer matching components. Thus, we can restrict comparison of components between $\boldsymbol{c}_P$ and $\boldsymbol{k}_L^{(\lambda\mu)}$ to the small subset of active components in $\boldsymbol{k}_L^{(\lambda\mu)}$. We even can restrict comparison to a subset of few hundreds of randomly selected active components $k_{nl}^{(\lambda\mu)}$ in order to get a quick first estimate. Recently, we have been investigating strategies based on the generalized Hough transform [9], but it is beyond the scope of this chapter to go into details. Finally, we should mention that multiple representation of $\boldsymbol{k}_L$ at different positions $(\lambda\mu)$ can be extended by additional representation at different orientations ψ. Of course, the search space between $\boldsymbol{k}_L^{(\lambda\mu\psi)}$ and $\boldsymbol{c}_P$ is again increased by one dimension but due to our previously discussed acceleration this is tolerable.

The combination of Gabor-based representations with the strategy of extrafoveal recognition allows recognition of objects also in real-world scenes with structured background. Although the search space can be reduced by foveation or by orientation estimation the quality of recognition does not depend on this support. On the other hand, the exact position and orientation of an object is available with high precision if the steps $(\lambda\mu\psi)$ of displacement and orientation in $\boldsymbol{k}_L^{(\lambda\mu\psi)}$ are reduced in the vicinity of the best match. Therefore, we could successfully use our recognition system for vision-guided disassembly of

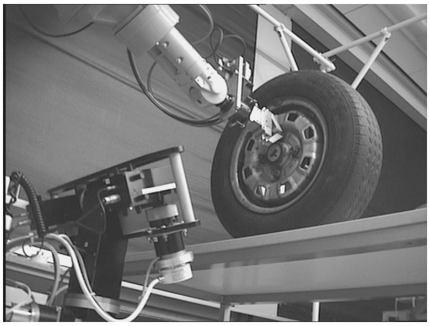


Figure 8.19: *A typical scene of our application in disassembling cars.*

used cars. Figure 8.19 shows a typical scene in which the type of wheel is recognized, the size of screws is identified, and the screw positions are measured in six degrees of freedom. Note that we have a pair of cameras in this application that allow additional depth measurements by stereo.

8.3.5 Advantages and limits of the holistic system

We began the discussion of our system with some observations of human visual behavior and we identified holistic and sequential recognition strategies. In the preceding sections we showed that rapid learning and robust recognition of objects can also be achieved by technical holistic systems. We also showed that learning of normalized representations in a foveating system provides position invariance, distance invariance and orientation invariance. We improved shape discrimination and robustness by a combination of binary or color-based representations with contour and vertex representations. Also, we achieved tolerance against inevitable foveation and normalization errors by tolerant representations, and these representations also happened to be tolerant against minor perspective distortions. Thus, we were able to extend our system to learning and recognition of 3-D objects. Further, we also improved the quality of our system under real-world conditions by introducing Gabor-based representations. Finally, we showed that the strategy of extrafoveal recognition no longer requires perfect segmentation.

However, two serious disadvantages remain and they can not be eliminated within the framework of a pure holistic approach. The first problem appears as soon as objects are considerably occluded. In this case foveation as well as orientation estimation will fail. Moreover, the strategy of extrafoveal recognition will provide subliminal similarity measures. The second problem is related to the tolerance-separability problem that was discussed in Section 8.2.3. On the one hand, we provide representations that tolerate errors in foveation as well as minor deviations in shape and minor perspective distortions. These representations suppress small details and even if they are represented, the recognition threshold will be adjusted too low in order to discriminate these details. However, it is also well known from human vision that we are not able to see all details at first sight. If we really want to see details we foveate them sequentially and shrink our visual field to see them with higher resolution. The strategy of sequentially foveating details and parts of objects will help us to overcome both problems, that is, the problem of occlusions and the problem of discrimination between details of similarly shaped objects. These sequential strategies seem to be knowledge-based in biological systems, and we also introduce a knowledge-based approach in the following section.

8.4 The hybrid neuro-artificial intelligence (AI) system

8.4.1 Advantages and limits of classical AI systems

During the last two decades, methods from the AI field have been widely used for computer vision. The principle idea is to build explicit object models of a domain. These models are typically based on a set of elementary features, such as straight or curved edges and lines, corners, areas of homogeneous intensity or color, etc. For the recognition process vision routines are needed to extract these elementary features. A control algorithm, often called an inference engine, is used to map the features to the object models. As a result, object recognition can be seen as a search or optimization problem that finds the best mapping M from a feature space to the elements of the object model. An introduction is given in Volume 2, Chapter 27.

When using semantic networks for modeling, we typically have a frame-like data structure of so-called concepts and a set of links between these concepts. The concepts are used to describe the properties of an object or its parts, while the links are used to model relations between the objects or their parts. Two standard relations that are often used in this field are part/part-of and the specialization/generalization relation. They form a hierarchy of abstraction with the most detailed model elements in the leaves of the network. Figure 8.20 shows a sample network of concepts and these standard relations.

Due to the explicit description of objects and their decomposition, partially occluded objects or disturbed objects can be recognized by finding a subset of their parts, that is, by finding subgraph isomorphisms [10, 11, 12]. Nevertheless, some major problems still appear. First, it is impossible to model all imaginable disturbances, and thus we rely on the robust extraction of the elementary features. Grouping techniques similar to those described in Witkin and Tenenbaum [13],

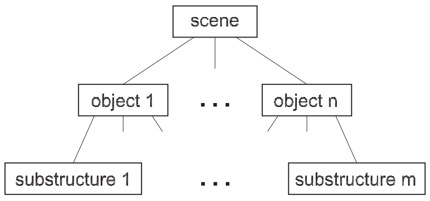


Figure 8.20: *Sample semantic network.*

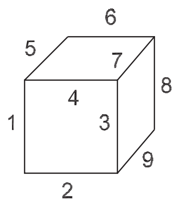


Figure 8.21: *Sample object.*

Lowe [14, 15], or Jacobs [16] are widely used to solve some of these problems. Second, a combinatorical explosion leads to large search spaces that have to be analyzed to find the best mapping M.

Consider the example in Fig. 8.21. A certain aspect of a cube is modeled by its nine straight lines and their topological relations (parallel to, perpendicular to, left of, right of, etc.). When mapping the image features to the object model, we have to consider 9^9 possibilities. This is done under the assumption of optimal extraction of the features without having other objects in the scene, without extracting features from the background, and without considering other object models in the model base. Obviously, one of the main problems is that we have to deal with an exponentially growing search space, especially when recognizing objects in real-world scenes.

Nevertheless, one of the main advantages of explicit object models is the handling of occluded objects by recognizing their details as mentioned earlier. We will show in the following paragraphs, how the major problems of conventional AI systems for computer vision can be overcome by modeling objects on the basis of holistically learned and recognized views of an object.

8.4.2 Modeling with holistically learned representations

As described in Section 8.3.5 the holistic neuro-recognition system is limited to objects that are not considerably occluded. In addition, holistic recognition reaches its limits when objects only differ in a few details. These limits can be overcome by combining the advantages of

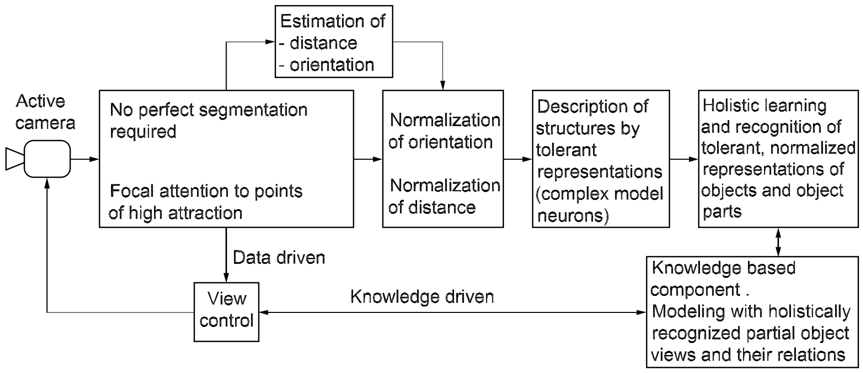

Estimation of
- distance
- orientation
Active camera
No perfect segmentation required
Focal attention to points of high attraction
Normalization of orientation
Normalization of distance
Description of structures by tolerant representations (complex model neurons)
Holistic learning and recognition of tolerant, normalized representations of objects and object parts
Data driven
View control
Knowledge driven
Knowledge based component .
Modeling with holistically recognized partial object views and their relations

Figure 8.22: *The holistic recognition system and classical AI are combined into a hybrid system.*

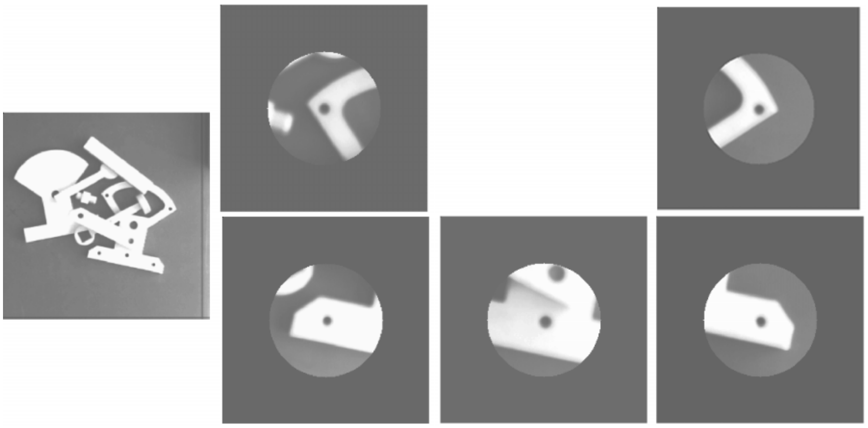


Figure 8.23: *Object view and partial views on some characteristic details.*

conventional AI systems with the robustness of holistic recognition in a hybrid system (Fig. 8.22).

In contrast to the classical approach, we model complex objects by partial or "narrowed" views, focusing on characteristic details of the objects [17]. These partial views are taken by moving the camera closer to interesting details, by pan/tilt-moves and by the use of the camera zoom lens, or by scaling mechanisms as described in Section 8.2.2. Figure 8.23 shows a sample of an occluded object and views to parts of the object focusing on characteristic details.

It can be seen that these partial views contain considerably more information than a simple elementary feature. We can outline the following advantages:

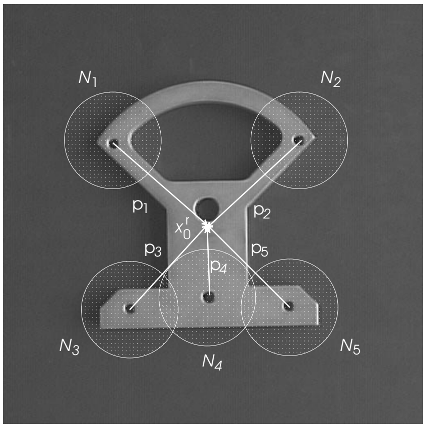


Figure 8.24: *Partial views and their reference vectors.*

1. While elementary features such as straight or curved edge segments of classical AI systems are shared between many objects, our feature "partial view" is object specific and therefore reduces the number of possible hypotheses, which have to be tested;
2. A partial view contains pose information, which allows estimation of the position of additionally modeled partial views. As a result, the search space can be reduced again; and
3. The integration of an explicit aspect hierarchy for active vision approaches enables the system to distinguish even between very similar objects.

Therefore, modeling with object-specific structures prevents a combinatorial explosion of the size of the search space.

For modeling the topological relations between different object views and several detailed views, we define an arbitrary but fixed reference point $\boldsymbol{x}_O^r = [x_O, y_O, z_O]^T$ (e.g., the center of mass or even better any point on the surface of the object) and a reference viewing direction $\boldsymbol{v}_O^r = [\alpha_O, \gamma_O, \varphi_O]^T$ (e.g., the surface normal at $\boldsymbol{x}_O^r$). We can now define a viewing point and a viewing direction for each detailed view $\boldsymbol{N}_i$ relative to $\boldsymbol{x}_O^r$, $\boldsymbol{v}_O^r$ by offset vectors $\boldsymbol{p}_i = [\boldsymbol{x}_i^{\text{offset}}, \boldsymbol{v}_i^{\text{offset}}]^T$.

Figure 8.24 visualizes several partial views $\boldsymbol{N}_i$ together with their offset vectors $\boldsymbol{p}_i$. As soon as one of these details $\boldsymbol{N}_i$ is recognized by our holistic system, we get information on its class name, its position $\boldsymbol{x}_i$, and its orientation $\boldsymbol{v}_i$ in the scene. We get a hypothesis of the pose of the whole object and its other details by transforming the camera coordinate system into an object-centered coordinate system by adding the appropriate offset vectors and by retransforming the coordinate

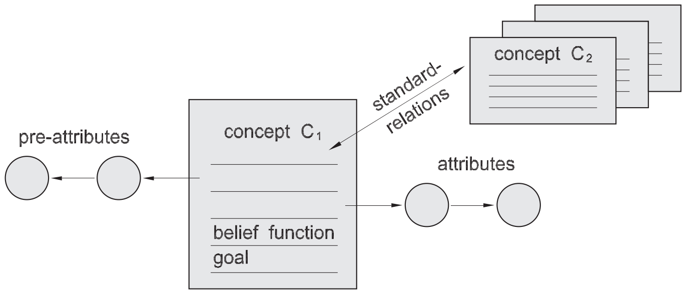


Figure 8.25: *A concept and its slots.*

system to the camera-centered one. Thus, after finding a first detail, all other details can directly be accessed.

After outlining the main ideas of modeling with holistically learned representations, we now describe several details of the semantic network language. An object model $(\mathcal{T}, \mathcal{R})$ of tolerant representations $\mathcal{T}$ and relations $\mathcal{R}$ can directly be mapped to a semantic network $S = (C, \mathcal{R}')$ of concepts C and relations $\mathcal{R}'$ by mapping $\mathcal{T}$ to C and $\mathcal{R}$ to $\mathcal{R}'$. We implement concepts as frame-like complex data structures. Several slots of each frame are used for the standard relations part/part-of, specialization/generalization to form a hierarchy of concepts. Several other slots are reserved to implement attributes of a concept. These attributes describe the properties of an object or a certain part of it (Fig. 8.25).

We use attributes not only to describe the properties of an object (declarative knowledge), but also to describe how these properties or features can be extracted from the image. A procedural interface implements the integration of computer vision routines in the semantic network (procedural attachment). We distinguish between two different sets of attributes. The first set, called pre-attributes, is used to model the holistic properties (holistic recognition, neural operators); the second set is used to model topological relations between the parts of an object, as it is common to knowledge-based approaches [18].

A belief slot is added to hold a belief function that models information on the reliability of the recognition. Several of these belief slots are accumulated by a Dempster-Shafer evidence mechanism. A goal slot is added for technical reasons and is used to guide the instantiation process as described in Section 8.4.3.

The modeling strategies that have been introduced in this chapter can obviously be extended to models of complex 3-D objects. Therefore, an aspect-of relation has been integrated into the semantic networks to include knowledge about characteristic 3-D views of an object. When modeling aspects of an object, we confine ourselves to a relatively

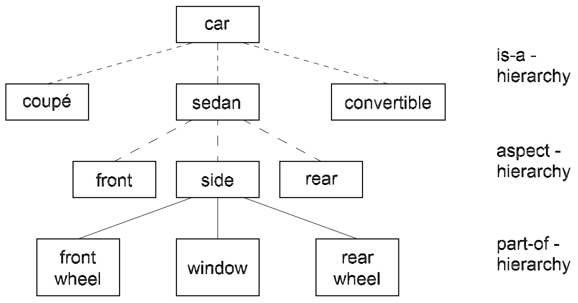


Figure 8.26: *Sample network of a complex 3-D object.*

small set of characteristic views of an object, as shown in Section 8.4.5. This distinguishes our approach from other aspect-based object models [19, 20, 21]. An example network for a 3-D object is given in Fig. 8.26.

Finally, we can summarize by stating that we consider an object model as a pair of characteristic views and relations between these views—the standard relations and some topological relations. According to our normalization procedures (Section 8.2.2), we do not store characteristic views as gray or color images but as normalized tolerant representations $\boldsymbol{t}_i$. Thus, we get a pair $(\mathcal{T}, \mathcal{R})$ of tolerant representations $\mathcal{T} = \{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n\}$ and relations $\mathcal{R} \subset \bigcup_{p \in N} \mathcal{T}^p$. Additionally, the procedural interface of the modeling language enables us to integrate knowledge to describe which vision routines should be used for extracting the representations. This includes also the necessary procedures for moving a robot and for grabbing camera images. Thus, we can model an active recognition process in our hybrid system.

8.4.3 Instantiation strategies

In the previous section, we have described how to model objects in our system by semantic and neural networks. However, an object model is only one part of a computer vision system. It has to be supplemented by a control algorithm that maps the image information to the elements of our symbolic object model and thereby instantiates the concepts of the semantic network. Due to the active vision approach of our system we do not only map the signal information of a single image to the network, but the information of a set of images is transformed into a symbolic description (Fig. 8.27). Furthermore, this set of images is dynamically grabbed during the mapping process.

In our case, an A*-based search mechanism is used to find the best mapping [22, 23]. Therefore, we must define a strategy that transforms the semantic network (and the image information) into a search tree

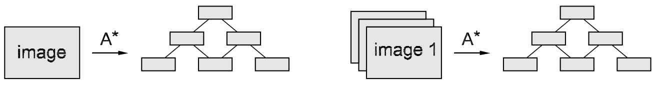


Figure 8.27: *Mapping image information to symbolic descriptions.*

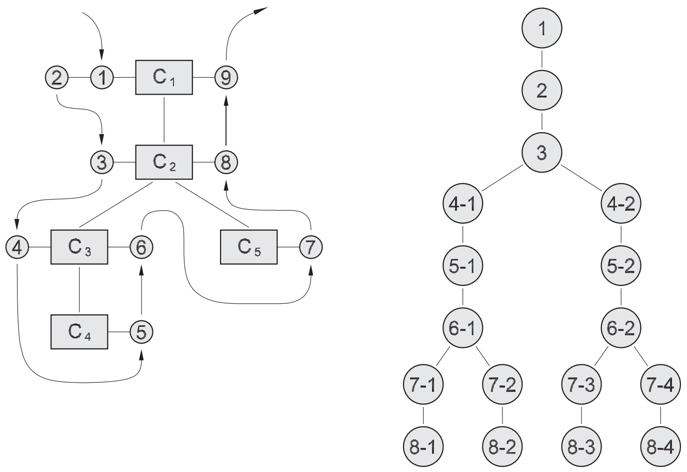


Figure 8.28: *Traversal of the network and a search tree.*

that can be evaluated by the A* algorithm. As explained earlier, the standard relations form a hierarchy of concepts with a single concept on top of the hierarchy, the root node of the hierarchical semantic network. For each concept, we use so-called pre-attributes to model the holistic recognition and attributes to model topological relations between substructures of an object. According to this structure of our networks we traverse the network as shown in Fig. 8.28, hence, the search tree is constructed dynamically.

Starting at the root node of the network, we traverse the network top-down the part/part-of hierarchy. When entering a concept C_i its pre-attributes are evaluated. These are used for the description of the holistic recognition process. In case of a successful holistic recognition, we change to the bottom-up instantiation phase, build a new instance of the concept and ascend in the hierarchy without evaluating the subnetwork below C_i. During the bottom-up phase the attributes of the concepts are evaluated. They are used to model the topological relations among parts of an object to ensure that the instantiated parts really form the superior object.

It can easily be seen that the structure of the semantic network and the traversal strategy determines the general structure of a path in the search tree. Due to the top-down traversal, the root node of the hierarchical semantic network becomes the root node of the search tree. Concepts that stay in a part-of relation to the same superior concept become successors in a left-to-right order. Concepts that are part of more than just one superior concept appear several times in a path of the search tree. Branches in the tree are created when the presented representation of an object matches different learned representations, which means when alternative mappings of image structures to the concepts appear. These alternatives are shown twice in Fig. 8.28 with two resulting branches each (pre-attribute no. 4 and attribute no. 7).

The belief function $b()$ is used during the instantiation phase to rate the chosen mapping. Typically, the distance measure $d(\boldsymbol{t}_L, \boldsymbol{t}_P)$ of the holistic recognition forms the main part of the belief function. In case of the decompositional recognition by evaluating the parts of a concept, the rating of each part, weighted by a reliability factor, forms another main part of the belief function. Several belief slots are accumulated by a Dempster-Shafer evidence mechanism that is not described in detail here [24]. The A* algorithm uses the belief function for the computation of its cost function.

The main rules for the instantiation process are summarized as follows:

1. A set of pre-attributes $\mathcal{A}_{\text{pre}}$ or attributes $\mathcal{A}$ of a concept C is calculated successfully if the value of the belief function $b(\mathcal{A}_{\text{pre}})$ or $b(\mathcal{A})$ of C is greater than a given threshold $b_{\min}$;
2. A concept C is instantiated if its pre-attributes $\mathcal{A}_{\text{pre}}$ are calculated successfully;
3. A concept C is instantiated if its attributes $\mathcal{A}$ are calculated successfully;
4. A subnet S with root node C is instantiated if the pre-attributes $\mathcal{A}_{\text{pre}}$ of C are calculated successfully; and
5. A concept C is instantiated if it is part of an already instantiated subnet S.

8.4.4 Recognition results

Experiments with our hybrid neuro-AI vision system have been made testing two different domains. First, objects of a robot benchmark, the Cranfield set, have been used to test the recognition of occluded objects in almost flat 2-D scenes. Figure 8.29 shows several scenes that could be analyzed successfully. Second, several toy cars have been modeled for the recognition of complex 3-D objects (Fig. 8.30).

For the second set of experiments, two toy cars (a Fiat Punto and a Ferrari F40) in the scale 1:24 have been used. The cars are colored

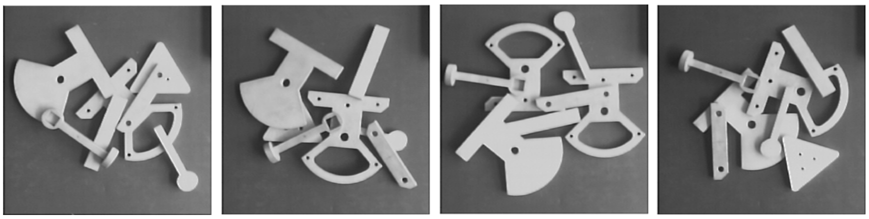


Figure 8.29: *Several scenes with occluded objects of the Cranfield benchmark.*

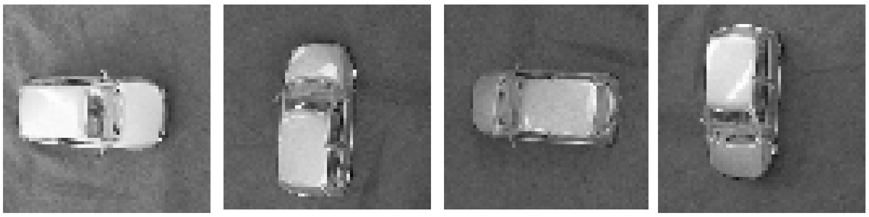


Figure 8.30: *Several holistically learned views of a car.*

and have very shiny surfaces. Their moderate size allows viewpoints within the relatively small reach of our robot. At the same time, they show many details that can be modeled in the semantic network and that can be analyzed by the recognition system. In the hybrid system, we are currently using the holistic system as described in Section 8.2. As we have not yet integrated the Gabor-based representation, the cars are presented to the vision system on a table with homogeneous background, and in view of the fact that the robot is placed close to a large-window side, the differing illumination conditions are still a problem for recognition. Illumination ranges from diffuse daylight to direct sunlight with shadows, etc., during the experiments. Resulting segmentation variations are cushioned by the tolerant-feature representation. Nevertheless, additional work has to be done for dealing with extreme shadows especially in the case of moving the camera against the light, which is definitely one of the most serious problems in active object recognition. In our experiments opposite light was a major difficulty when recognizing the rear view of the cars.

Table 8.1 shows the recognition rates for the objects as well as for some of their details that have been used in the models. Obviously, there are several misclassifications and rejections of some partial views. Nevertheless, the whole recognition rate increases when using the complete model information. Especially, the rate of misclassifications clearly was reduced. The toy cars were presented to the recognition system 100 times in different positions and under varying lighting conditions.

Table 8.1: *Recognition rates for objects and some of their details*

Object "Fiat"	whole	front	left side	right side	rear
Correct classification	88 %	76 %	88 %	86 %	68 %
Misclassification	0 %	10 %	0 %	0 %	14 %
Rejection	4 %	4 %	4 %	10 %	8 %
Unreachable viewpoint	-	10 %	8 %	4 %	10 %
Generalization "car"	8 %				
Object "Ferrari"	**whole**	**front**	**left side**	**right side**	**rear**
Correct classification	80 %	86 %	82 %	76 %	72 %
Misclassification	0 %	4 %	10 %	20 %	20 %
Rejection	0 %	0 %	0 %	0 %	0 %
Unreachable viewpoint	-	10 %	8 %	4 %	8 %
Generalization "car"	20 %				

In addition to the Fiat and the Ferrari models, a generalizing concept "car" was used in the semantic network to model a general description of a car without including the typical details of a certain car type. It can be seen that in almost every case in which a recognition of the concrete car type was impossible, this generalizing concept was instantiated. The table also includes data on the rate of unreachable viewpoints for some of the aspects and details.

8.4.5 Automatic generation of object models

For a computer vision system to be effective, it is very essential to integrate learning strategies to easily adapt it to different objects and domains. While learning on the neural, subsymbolic level is done by a "transform and store" mechanism as described in Section 8.2.5, learning on the symbolic level is currently done by selecting a set of characteristic views of an object that are integrated into the semantic network. To restrict the problem of actively learning models, we first define some constraints:

1. During the learning phase objects are presented under "good" conditions. This is due primarily to the one-step unsupervised learning mode;
2. The robot holding the camera can reach all of the viewpoints, which are calculated during the learning phase. Obviously, the characteristic views can only be learned if the appropriate images can be taken. However, for recognition not all of the views have to be reachable;

3. Object classes shall be defined by similar shape of objects, not by similar function. A supervisor would be necessary to define classes of objects with a similar function.

In the following section we will concentrate on the selection of characteristic views of an object.

Characteristic views. When modeling objects in a semantic network by different views, the main question that has to be answered is: which aspects of an object and which of its details are especially suitable for unambiguous recognition of complex 3-D objects? Therefore, we do not try to find a set of views that allows a detailed reconstruction of the object nor are we looking for a nonredundant set of views. Instead, we are seeking a small set of views that is useful for discriminating between different objects. This distinguishes our approach from other work in the field of characteristic view selection [25, 26, 27]. For our task, we extract a set of possible candidates with our subsymbolic vision routines. These candidates are determined by foveation techniques that extract points of symmetry, regions of homogeneous color or intensity, and corners. The set of images we get when actively focusing the camera on these foveation points is transformed to the set of corresponding feature vectors. As mentioned earlier, these feature vectors are binary vectors, each element representing the activity of one model neuron. In a second step we use our subsymbolic classifier to determine how often these feature vectors may be misclassified. Only those feature vectors that are seldom misclassified are considered as being characteristic for a certain object.

In the third step, these feature vectors are used for modeling the objects in the semantic network. For each feature vector it is known from which position and viewing direction the corresponding image was taken. These absolute camera positions are used to compute the relative positions between the characteristic views, which are then stored in the semantic network as topological relations. In addition, for each feature vector it is known which vision routines were used for its computation. These steps—image acquisition, image preprocessing, feature extraction, and classification—are stored symbolically in the slots of the concepts of the semantic network. The global view on an object and the extracted views on characteristic details are connected by the standard part-of relation in the semantic network. Several global views on the object from different viewpoints are connected by the aspect-of hierarchy in the network. A formal description of the selection of the characteristic views is given next. Therefore, we first consider the 2-D case and the extraction of the part-of hierarchy.

Let us first consider a given global 2-D view, an aspect $\mathcal{A}$ of an object $\mathcal{O}$. Let $\mathcal{F}(\mathcal{A}) = \{f_1, f_2, \dots f_n\}$ be the set of foveation points in this view, which are points of interest that can be used for detailed views.

Let $\mathcal{N}(\mathcal{A})$ be the set of normalized images we get when actively focusing the camera on these foveation points and let $\mathcal{T}(\mathcal{A}) = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \dots, \boldsymbol{t}_n\}$ be the set of corresponding feature vectors. As with the holistic recognition level, we use the similarity measure $d(\boldsymbol{t}_i, \boldsymbol{t}_j)$, which is based on the inner product and the Hamming distance of the feature vectors, to approximate the probability that $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$ belong to the same class of objects by $d_{ij} = d(\boldsymbol{t}_i, \boldsymbol{t}_j)$. The probability matrix $\boldsymbol{D} = [d_{ij}]$ gives us several pieces of information on the substructures of the aspect $\mathcal{A}$ that are represented by the feature vectors. Therefore, consider the unit step function $u : \mathbb{R} \to \{0, 1\}$ with $u(x) = 1$, if $x > 0$; and $u(x) = 0$, if $x \leq 0$.

Let $\boldsymbol{S} = [s_{ij}] \in \{0, 1\}^{nxn}$ be a matrix with $s_{ij} = u(d_{ij} - d_{\min})$ and a given threshold $d_{\min}$. For our experiments, we use $d_{\min} = 0.8$. Let $\boldsymbol{s}_i = (s_{i1}, s_{i2}, \dots s_{in})$ with the vector norm $\|\boldsymbol{s}_i\|_1 = \sum s_{ij}$. Obviously, at least the diagonal elements s_{ii} are equal to one and thus we get $\|\boldsymbol{s}_i\|_1 \geq 1$.

We can now distinguish three types of feature vectors $\boldsymbol{t}_i$:

1. $\|\boldsymbol{s}_i\|_1 = 1$: $\boldsymbol{t}_i$ does not belong to any other classes; $\boldsymbol{t}_i$ contains information on the orientation and position of the object $\mathcal{O}$ (compare Section 8.4.2), because it is hardly misclassified;
2. $1 < \|\boldsymbol{s}_i\|_1 < s_{\max}$ with a given threshold $s_{\max}$: $\boldsymbol{t}_i$ should not be used to calculate orientation and position of the object due to possible misclassifications. Nevertheless, $\boldsymbol{t}_i$ can be used for testing the hypothesis that an unknown object is of the same class $C(\mathcal{O})$ as $\mathcal{O}$, because there are only a reasonable number of possible misclassifications. Additionally, its influence is decreased by a lower value in the evidence slot of the network notation (we use $s_{\max} = n/10$);
3. $\|\boldsymbol{s}_i\|_1 \geq s_{\max}$: $\boldsymbol{t}_i$ should not be used for modeling the object due to the large number of misclassifications.

Extension to the automatic generation of three-dimensional models. Obviously, we can generalize the earlier described procedure to views of multiple objects $\mathcal{O}_k$. Thereby, we get those characteristic views, which are useful for discriminating between the objects. In this case, the objects are modeled in the semantic network by views of details in the part-of-hierarchy. For 3-D objects, of course, we have to consider different aspects $\mathcal{A}_{kl}$ of each object. Therefore, a set of initial aspects of each object is obtained by sampling the viewing sphere with a constant step size, which can be user-defined. The characteristic aspects are then integrated in the semantic network by the aspect-of-relation.

Constructing semantic networks. As described in the foregoing, the topological arrangement of the selected views, the corresponding computer vision routines, and the necessary robot movements are stored in the semantic network. The computer vision routines that were used for grabbing images, foveation, feature extraction, and classification as well as the routines responsible for moving the robot with the camera on its hand are stored together with the necessary parameters in several slots of the concepts. Additionally, the learning process is able to categorize views and to combine similar views into a single concept of the network. Therefore, the definitions of a concepts classification attribute is extended by a set of possible outcomes of an operation. Semantic networks already have been automatically generated for both domains that were introduced in Section 8.4.4—the Cranfield benchmark and 3-D toy cars. Both have been successfully used for recognition tasks.

8.5 Conclusion

In this chapter we not only gave a detailed description of our hybrid Neuro-AI system, but we also tried to show its philosophy and our arguments in favor of the realized architecture. Hence, there is no need for a concluding recapitulation. Instead we give some outlook on future improvements.

When we introduced Gabor-based representations in Section 8.3.3, we improved the representation of contour structures in blurred low-contrast images of real-world scenes. Simultaneously, we replaced our original algorithm Eq. (8.8) and its tolerant representations by a simplified algorithm Eq. (8.16) providing Gabor-based tolerant representations. Therefore we lost the feature of "combinatorical filtering" by which essential structures were preserved and learned while noise and textures were removed. There are serious suggestions that biological systems also are able to separate essential visual structures by synchronization-based dynamical feature linking and this mechanism seems to be very similar to our original algorithm Eq. (8.8). Thus in future work we will combine our old linking mechanism with Gabor-based representations.

In Section 8.4.2 we showed how to model objects by a set of partial views and their mutual relations. Generally, the description of these relations contains geometrical information and in some cases these geometrical relations can be interpreted as positions of object parts within a more global view. This allows an alternative strategy for nonexplicit modeling. Extrafoveal recognition (Section 8.3.4) provides probability distributions for positions of holistically recognized object parts. These 2-D probability distributions can be matched with previously learned actual positions of parts. We are currently investigating whether this strategy of hierarchical grouping is able to replace lower levels of explicit modeling.

We also should mention that Gabor-based representations are only implemented in the holistic system and are not yet available for explicit modeling. Integration of the AI module with the more advanced Gabor-based system will improve the performance of the hybrid system and will not present major problems. The only problem arises with automatic learning of object models. Up to this point, we have selected partial views by foveation and evaluated a subset of significant prototypes. This strategy is to be extended to the case of real-world scenes where foveation is difficult or impossible. However, in spite of all these pending modifications and in spite of unpredictable problems we are sure to have a good basis with our hybrid AI-Neuro system.

8.6 References

[1] Hartmann, G., (1991). Hierarchical neural representation by synchronized activity: A concept for visual pattern recognition. In *Neural Network Dynamics*, J. Taylor, E. Caianiello, R. Catterill, and J. Clark, eds., pp. 356–370. London: Springer.

[2] Hartmann, G., Drüe, S., Kräuter, K., and Seidenberg, E., (1993). Simulations with an artificial retina. In *Proceedings of the World Congress on Neural Networks 1993*, Vol. III, pp. 689–694. Hillsdale, NJ: Lawrence-Erlbaum.

[3] Hubel, D. and Wiesel, T., (1959). Receptive fields of single neurons in the cat's striate cortex. *Jour. Physiology*, **148**:574–579.

[4] Hartmann, G., (1983). Processing of continuous lines and edges by the visual system. *Biological Cybernetics*, **47**:43–50.

[5] Hartmann, G., (1987). Recognition of hierarchically encoded images by technical and biological systems. *Biological Cybernetics*, **57**:73–84.

[6] Hartmann, G. and Drüe, S., (1994). Why synchronization? An attempt to show quantitative advantage. In *Proceedings World Congress on Neural Networks 1994*, Vol. I, pp. 581–586. Hillsdale, NJ: Lawrence-Erlbaum.

[7] Hartmann, G., (1991). Learning in a closed loop antagonistic network. In *Proceedings of the ICANN-91*, T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, eds., pp. 239–244. Amsterdam, North Holland: Elsevier.

[8] Dunker, J., Hartmann, G., and Stöhr, M., (1996). Single view recognition and pose estimation of 3D objects using sets of prototypical views and spatially tolerant contour representations. In *Proceedings of 13th Conference on Pattern Recognition*, Vol. IV, pp. 14–18. Los Alamitos, CA: IEEE Comp. Society Press.

[9] Ballard, D., (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, **13(2)**:111–122.

[10] Bunke, H. and Messmer, B. T., (1995). Efficient attributed graph matching and its application to image analysis. In *Proc. of the 8th ICIAP' 95*, pp. 45–55. Berlin: Springer.

[11] Ullman, J., (1976). An algorithm for subgraph isomorphism. *Jour. ACM*, **23(1)**:31–42.

[12] Wilson, R. and Hancock, E., (1994). Graph matching by configurational relaxation. In *Proceedings of the 12th ICPR*, Vol. B, pp. 563–566. Los Alamitos, CA: IEEE Computer Society Press.

[13] Witkin, A. and Tenenbaum, M., (1983). On the role of structure in vision. In *Human and Machine Vision*, A. Rosenfeld and J. Beck, eds., pp. 481–543. New York: Academic Press.

[14] Lowe, D., (1985). *Perceptual Organization and Visual Recognition*. Boston: Kluwer Academic Publishers.

[15] Lowe, D., (1987). Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, **31**:355–395.

[16] Jacobs, D., (1989). Grouping for Recognition. 1177, M.I.T. AI Lab Memo.

[17] Büker, U. and Hartmann, G., (1996). Knowledge-based view control of a neural 3-D object recognition system. In *Proceedings of 13th Conference on Pattern Recognition*, Vol. IV, pp. 24–29. Los Alamitos, CA: IEEE Comp. Society Press.

[18] Niemann, H., Sagerer, G., Schroder, S., and Kummert, F., (1990). ERNEST: A semantic network system for pattern understanding. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **12**:883–905.

[19] Dickinson, S., Christensen, H., Tsotsos, J., and Olofsson, G., (1997). Active object recognition integrating attention and viewpoint control. *Computer Vision and Image Understanding*, **67(3)**:239–260.

[20] Koendering, J. and van Doorn, A., (1979). The internal representation of solid shape with respect to vision. *Biological Cybernetics*, **32**:211–216.

[21] Munkelt, O., (1995). Aspect-trees: generation and interpretation. *Computer Vision and Image Understanding*, **61(3)**:365–386.

[22] Nilsson, N., (1980). *Principles of Artificial Intelligence*. Palo Alto, Ca: Tioga.

[23] Pearl, J., (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, Mass.: Addison-Wesley.

[24] Shafer, G., (1976). *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton University Press.

[25] Gremban, K. and Ikeuchi, K., (1994). Planning multiple observations for object recognition. *Int. Journal of Computer Vision*, **12(2/3)**:137–172.

[26] Hlavac, V., Leonardis, A., and Werner, T., (1996). Automatic selection of reference views for image-based scene representations. In *Proceedings 4th ECCV '96*, pp. 526–535. Berlin: Springer.

[27] Kutulakos, K. and Dyer, C., (1994). Recovering shape by purposive viewpoint adjustment. *International Jour. Computer Vision*, **12(2/3)**:113–136.

9 Active Vision Systems

Bärbel Mertsching and Steffen Schmalz

AG Informatikmethoden für Mikroelektronikanwendungen
Universität Hamburg, Germany

9.1 Introduction

Up to the end of the 1980s the most influential *computational theory of vision* was that developed by Marr which dominated the computer vision community. His approach was the first to articulate a comprehensive vision theory in computational terms, an enormous contribution with many different aspects. Vision was regarded as a reconstruction process, that is, leading from 2-D images through the primal sketch and the 2 1/2-D sketch to object-centered descriptions (*from pixel to predicates*, as expressed by Pentland [1]). Though much effort was invested to improve the methods, computer vision seemed to have reached a dead end. A comparison of the worldwide range of academic research activities and the status of the integration of its results in applications revealed:

- only simple and highly specialized problems were addressed;

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

- legitimate demands concerning the robustness when not exactly recordable conditions are prevailing, and adaptability to similar domains or real-time performance were only seldom fulfilled; and
- academic research neglected the dominant system character of a technical solution for an application.

While the application of visual sensors for measuring tasks showed little discrepancies (physics and signal theory provide an expandable repertoire of methods), a theoretical background for image analysis was still missing. Computer vision needed more than what image processing, pattern recognition, and sensor physics offered.

A solution emerged as Bajcsy argued that the problem of perception was not necessarily one of signal processing but of control of data acquisition [2, 3]. This solution was obtained from an examination of human perception, which is not passive but active. *"We don't just see, we look"* [3, p. 996]. While in passive vision systems the camera providing the image input is immobile, human perceptual activity is exploratory and searching. In the process of looking, humans move their heads to obtain a better view of the scene, and their eyes focus on certain items and adapt to different viewing conditions. An emulation of biological vision processes by technical systems promised a new direction in computer vision. As Aloimonos pointed out, problems that are ill-posed for a passive observer became well-posed and linear for an active observer. It is important to notice that active sensing can be performed with passive sensors. At the end of the 1980s the pioneering articles of Aloimonos et al. [4] and Bajcsy [3] established the *active vision paradigm*. From the beginning, it became perfectly clear that active vision required real-time information processing and control. Fortunately, progress in the availability of dedicated and powerful computer systems and image processing hardware that occurred in parallel made such approaches possible.

In this chapter we pursue active vision in a essayistic manner from the beginning to recent developments. In the following, we examine the essential ideas of Marr's theory and afterwards present criticism and conclusions. Next, the basic concepts of active vision systems are introduced and discussed in some detail. There is a great potential for further advances in these areas. Therefore, the state-of-the-art and proposals to further it are described in Sections 9.4 and 9.5 followed by a conclusion.

9.2 Marr's theory and its drawbacks

The work of Marr had and still has large impact on research in computational vision, human perception, and also human cognition [5, 6]. Its influence is easy to understand: his ideas, especially in his book *Vision*, are presented clearly and succinctly. In a relatively thin volume, Marr has packed more theory, and explained it more clearly, than many other authors in far larger volumes. In addition, his ideas simplify theorizing about perception, and provide a firm base for model building. He gave a complete description of processes relevant for vision, starting with early visual processing up to 3-D model representation and object recognition. In addition, he attributed importance to the biological plausibility of his algorithms. In the following the main principles of Marr’s theory are summarized in order to explain its drawbacks as an overall framework for studying and building image analysis systems.

9.2.1 Principles

Marr formulated his ideas on the basis of the information processing paradigm of cognitive science using the physical symbol system hypothesis of Newell and Simon [7]: vision as well as other cognitive capabilities can be formalized as a pure information processing task and can therefore be founded on a general theory. This implied a closed framework for the treatment of visual tasks and the existence of general solutions not depending on special implementations. He assumed further that information exists in the world independently of a processing system and that it has only to be collected or sampled.

According to Marr, the top-down design of visual systems consists of precise mappings of one information entity to another on a higher abstraction level. In this way, he suggested three levels *“at which an information-processing device must be understood before one can be said to have understood it completely”* [6, p. 24]:

- a computational theory for the formulation of a general task and its solution based on necessary constraints and boundary conditions;
- a specification of algorithms and a representation scheme providing a solution of formal procedures; and
- a hardware implementation.

These three levels form the basis for the understanding of information processing. They are logically and causally related but can be used independently for explanatory purposes.

The representation hierarchy occupies an important space in Marr’s theory. Starting with intensity values in images, he first specified the *primal sketch* that includes explicit information such as zero-crossings, edges, grouped similar symbols with their boundaries, and virtual lines. Marr stated that the human visual system detects edges by evaluating the zero-crossings of the Laplacian of a Gaussian-filtered image [8], which he approximated through the difference of two Gaussian functions. The goal of the primal sketch is to detect so-called tokens and their boundaries. Therefore, discrimination processes are necessary to separate similar tokens and to form boundaries between them. Furthermore, grouping processes are used to combine and build up larger tokens. Following the primal sketch Marr emphasized the role of intrinsic images (the *2 1/2-D sketch*) for use in what became known as *early vision* in order to add information about orientation and rough depth. Support for these representations has been found in the retinotopic maps of early vision (optic flow, texture, color, disparity). Discontinuities in these quantities are analyzed and stored in a viewer-centered representation. Finally, he formulated a 3-D model representation to describe object shapes and their spatial relations using a hierarchical object-centered representation including volumetric and surface primitives: *"To construct a 3-D model, the model's coordinate system and component axes must be identified from an image, ..."* [6, p. 313]. Marr used so-called component axes to describe object parts. These axes are given in a viewer-centered coordinate system and therefore a transformation, the image-space processor, is necessary to represent entities in an object-centered system. The recognition process consists of two modules: *"a collection of stored 3-D model descriptions, and various indexes into the collection that allow a newly derived description to be associated with a description in the collection"* [6, p. 318]. These collections he refers to as the catalog of 3-D models that is hierarchically organized:

1st level. It contains the most general descriptions providing information only about size and overall orientation.

2nd level. Here information about the number and distribution of axes of the components is included allowing a rough determination of shape configurations.

3rd level. Finally, a more accurate description of the sizes of the components and the angles between them is derived.

Accordingly, the recognition and formation of new models occur within this hierarchy: *"A newly derived 3-D model may be related to a model in the catalogue by starting at the top of the hierarchy and working down the levels through models whose shape specifications are consistent with the new model's ..."* [6, p. 320]. Marr defines three access paths into the catalog:

Specificity index. Marr's notion of recognition rests upon the specificity index that concludes the hierarchical top-down recognition.

Adjunct index. Once a 3-D model is selected it provides 3-D models of its components.

Parent index. If a component of a shape was recognized the parent index provides information about the type of the whole shape.

Marr points out that the recognition process may be ambiguous. But using contextual constraints and other available clues such as thickness of shapes, symmetries, repetitions, and large differences in angles, he wants to overcome these problems.

9.2.2 Criticism

After an enthusiastic reception to the book *Vision* in the first years after its publication, the limitations became obvious. Computer vision had long been discussed in the context of artificial intelligence and knowledge-based systems with the result that the processing of visual information consisted of the same methodology: a syntactic analysis followed by symbolic processing of the results. Most work involved extensive static analysis of passively sampled data. This concept was very well suited to synthetic AI problems as there are block worlds, Lambertian surfaces, smooth contours, controlled illumination, etc., but failed when used to develop general useful and robust systems. All working image processing systems available today are only capable of solving particular tasks due to the limitations of system resources. While Marr's theory promoted the formalization of connections between macroscopic phenomena of the world (spatiotemporal structures) and the ability of humans to perceive them, it lacked the practical application of its insights. Marr thought of the human visual system as an well-defined, closed information processing device that implied the possibility of an exact mathematical specification and a strictly separated treatment of all processes. He defined visual perception as a reconstruction process of the sensory input, that is, perception is an inversion of a mapping—an inversion problem, which is only solvable under the forementioned well-defined conditions and assumptions and therefore an ill-posed problem. Research during the last decades has shown that a general formalization of visual perception is not possible. Moreover, the hierarchical organization of Marr's representation scheme prevented the introduction of learning, adaptation, and generalization.

Despite the major contributions of Marr, his theory left major issues unaddressed. First, it neglected the perceiver's behavior and was therefore essentially about passive vision. Moreover, special features of the human visual information processing, such as the elaborate gaze control system and the changing resolution of the retina from the fovea to the periphery, were neglected. In addition, Marr did not take into account the importance of additional knowledge sources available within biological visual processing systems. Instead he wrote: *"even in difficult circumstances shapes could be determined by vision alone"* [6, p. 36]. Especially, the analysis of complex scenes with distant objects requires knowledge about these objects, their arrangements, and possible occlusions due to the insufficient disparity of information for such objects to yield depth representations. Generally speaking, Marr stressed too much the representation and the recognition capabilities: *"all other facilities can be hung off a theory in which the main job of vision was to derive a representation of shape"* [6, p. 36]. His concentration on object-centered representations for 3-D object recognition is questioned by newer research results that postulate a viewer-centered representation following psychophysical and neurophysiological investigations [9, 10, 11].

Most of all, Marr saw vision as a general process independent from the observer and the particular visual task. Aloimonos' criticism is as follows: *"Vision was studied in a disembodied manner by concentrating mostly on stimuli, sensory surfaces, and the brain"* [12]. While Marr analyzed the entire image, that is, he wanted to collect all computable information for all parts of a input scene, biological vision systems build up selective representations about the world in conjunction with their actions. Furthermore, Marr neglected that visual information processing is a context-sensitive process: only certain information sources are needed for a particular vision task. Thus, the processing of complex scenes with ambiguous information was beyond the range of Marr's theory.

All these observations of the drawbacks of the computational theory of vision led to several conclusions for further research: general-purpose vision has proven to be a chimera. There are simply too many ways in which image information can be combined, and too much that can be known about the world for vision to construct a task-independent description. Moreover, it is obvious that vision does not function in isolation, but is instead part of a complex system that interacts with its environment. It seems useful to extract information that is not readily available from static imagery by direct interaction with the physical world. In order to make an economical use of the resources it will not be possible to accumulate a maximum of knowledge. On the contrary, the world can serve as a database with the vision system retrieving task-oriented the relevant information by directing gaze or attention. Behavioral vision demands adaptability, implying that the functional characteristics of the system may change through interaction with the world.

9.3 Basic concepts of active vision

The redefinition of the paradigmatic base of computer vision as mentioned in the foregoing started about one decade ago and is yet ongoing. Several authors contributed to the contemporary spectrum of alternative paradigms and emphasized different aspects that are reflected in the terms they used to characterize their approaches. They all have in common that they regarded the visual system as a sensory, perceptual, and motoric one. Another common assumption is the inclusion of feedback into the system and the gathering of data as needed.

Active vision. The active vision paradigm [4, 13, 14] has tried to overcome some of the forementioned drawbacks of Marr's theory by the introduction of an active observer. *"An observer is called active when engaged in some kind of activity whose purpose is to control the geometric parameters of the sensory apparatus. The purpose of the activity is to manipulate the constraints underlying the observed phenomena in order to improve the quality of the perceptual results"* [13, p. 140]. The active control of the outer degrees of freedom of the ocularmotor system enables a vision system to break down the complexity of a task [15, 16, 17]. Later, Aloimonos focused on the *purpose* as the driving power of a system to interact with its environment (*purposive vision*, [18]) and the efforts in realizing any visual task (*qualitative vision*, [19]).

Animate vision. The term from Ballard [20] shall focus on the human visual system as a complex repository of technical insights on how to use vision in the course of answering a huge repertoire of questions about the world: *"... researchers ... seek to develop practical, deployable vision systems using two principles: the active, behavioral approach, and task-oriented techniques that link perception and action"* [21]. Similar to active vision animate vision also supports the division in preattentive phases (fast and data-driven bottom-up processing) and attentive stages (top-down controlled and related to the next task) [21, 22].

Active perception. Bajcsy stressed the combination of modeling and control strategies for perception: *"... <active perception> can be stated as a problem of controlling strategies applied to the data acquisition process which will depend on the current state of the data interpretation and the goal or the task of the process"* [3, p. 996]. In order to accomplish closed feedback loops it is necessary *"to define and measure parameters and errors from the scene which in turn can be fed back to control the data acquisition process"* [3, p. 1004]. In further publications she introduced the concepts of exploratory perception and active cooperation [23, 24].

In the following, we use a broad definition and summarize all these approaches using the general term *active vision*. Swain and Stricker [16, p. ii] give the following characterization of active vision systems, which is still current:

"Active vision systems have mechanisms that can actively control camera parameters such as position, orientation, focus, zoom, aperture

and vergence (in a two camera system) in response to the requirements of the task and external stimuli. They may also have features such as spatially variant (foveal) sensors. More broadly, active vision encompasses attention, selective sensing in space, resolution and time, whether it is achieved by modifying physical camera parameters or the way data is processed after leaving the camera.

In the active vision paradigm, the basic components of the visual system are visual behaviors tightly integrated with the actions they support: these behaviors may not require elaborate categorical representations of the 3-D world."

It is obvious that the term *active vision* may not be confused with active sensing (sonar sensors for the measurement of distances, laser systems for the generation of depth maps, etc.).

Such a broad definition of the field requires research in areas that had not been studied before: Due to the highly redundant source *image* signal *attention* has to be considered for active vision systems with highest priority in order to achieve figure-ground separation and real-time performance. Attention means selective processing of volumes-of-interest within the system environment with restricted size and motion and with respect to the specific task. The signal selection may take place along various signal dimensions: distance; space; and velocity. *Gaze control* is needed to direct a system's attention to different interesting locations in its environment. Parameters of cameras and joints of camera-heads have to be altered to perform gaze shifts. It is obvious that new *hardware concepts* for integrated camera platforms and image processing hardware are indispensable. If sensors are connected to robot manipulators, the system can acquire additional information, determining material, geometric and haptic characteristics of objects. Such an integration with robot architectures opens the door to an intensive cooperation with the robotics community. While robot systems normally use accurate calibrations, an *eye-hand coordination* should work without facilitating the cooperation of the different subsystems. Attention, gaze control, hardware and software concepts and eye-hand coordination are discussed in more detail in the following sections.

Further research themes have be considered under a different light: Spatially variant sensors allow *foveal sensing* providing high resolution at the location of interest without the cost of uniformly high resolution as well in the periphery [25, 26, 27]. Spatial coordinate transformations (e.g., log-polar) support this property [28, 29]. A typical active vision application will likely consist of many computationally intensive tasks, each benefiting possibly from *parallel processing*. These programs need an underlying software system that not only supports different parallel programming models simultaneously, but also allows tasks in different models to interact [21]. Though in the active vision paradigm space-time representations are used to serve the needs of specialized units devoted to a number of tasks and therefore elaborated 3-D representations of the world may not be necessary, the interface to high-level tasks as *object recognition* and *scene interpretation* should not be neglected. But algorithms that recognize objects and events should be stable to large amounts of noise and should be executable in real-time.

Vision and in particular active vision should not be studied in isolation but in conjunction with other disciplines. In areas such as neurophysiology and molecular biology new techniques have been developed that allow tracing the visual information processing in living beings at the molecular, neural, and cellular levels. Behavior as associated with active vision systems can be observed by empirical disciplines such as psychophysics, cognitive psychology, and ethology. By now, some insight is gained concerning the various functional components of the brain that can be used to emulate the biological vision system. In the ideal case, computational neuroscience renders possible modeling at different levels in a way that a biologically adequate model at a particular level is derived, whereas empirical facts from neighboring levels constrain the space of potential models for that particular level.

9.3.1 Attention

Various authors use the metaphor *spotlight* in order to describe the phenomenon *attention*. Such a spotlight moves steadily from one position to another slipping over regions that lay between a starting and a goal position. The spotlight is assumed to be a homogeneous compact spot of a certain size. But the main prerequisite for the spotlight hypothesis is that attention cannot be allocated simultaneously for different, spatially separated regions of the visual field [30, 31]. A second term for spotlight is *focus of attention*.

Directing the gaze of an exploratory vision system by a *visual attention* model has become a useful paradigm in active vision, also called *attentive vision* [32]. Visual attention reduces the amount of data to be processed per time step and serializes the flow of data. It facilitates the spatial representation and simplifies mathematics by producing reference points and reducing the degrees of freedom. At least four mechanisms are known for data reduction in early visual processing: *Signal selection* is used for focusing on interesting regions and events, that is, to select critical segments of the signal for further processing while ignoring the majority of it that is not relevant. While *data compression* seeks to eliminate the redundancy in the signal, *alerting* determines where to look next. Certain *property measurements* can be computed directly on sensor data (e.g., component motion and depth).

The attention model by Koch and Ullman [33] is a basic contribution to the development of computational attention models. Though they did not implement it, it contains some important demands that can be found in several later models. It possesses the following features:

- There is a parallel representation of elementary object features (e.g., color, orientation, direction of motion) in different feature maps.
- A mapping exists that projects the topographic maps onto a central nontopographic representation. This map consists of one single point, describing where to look next.
- A winner-takes-all network derives the localization of the most interesting attention point.
- After a gaze shift to an interesting point, an inhibition of this point occurs, releasing a shift to the next attention point.
- Additional rules may define adjacency relations.

Computational attention models can be divided into two major classes. Most of the connectionist-based attention models are concerned with the transformation of an object (or a similar structure) into an object-centered reference frame (e.g., [34, 35, 36]). These routing mechanisms often imply not only a position, but also a size-invariant mapping. In an active vision system both effects can also be achieved by the control of the camera platform (pan-tilt, vergence) or the cameras themselves (zoom, focus), respectively. Other models deal with the question of how a salient region can actually be found and what region may be relevant to the current task (e.g., [37, 38]). Such a strategy seems to be promising for the gaze control of an attentive vision system.

9.3.2 Gaze control

In order to analyze *gaze control*, the problem can be partitioned according to biological mechanisms that drive eye movements, such as vergence and smooth pursuit. Neither vestibulo-ocular phenomena (e.g., the compensation of egomotion), nor elaborate visual processes have been modeled yet. Unfortunately, the cooperation between these modules and their integration still remains insufficiently investigated. Thus, a simpler approach is the division of the task in two categories: *gaze stabilization* or *fixation* maintains the optimal view of a particular interesting object that may be stationary or in motion with respect to the camera while *gaze change*, similar to human saccadic eye movements, deals with transferring the gaze for exploring an environment, recognizing parts of objects, and tracking moving items.

The primary goal of gaze control is the active manipulation of camera parameters so that the images acquired are directly suited to the task specified. The parameters include the six degrees of freedom for the camera position, lens parameters such as aperture, focus, zoom, plus relative parameter values for multicamera systems. In contrast to the computational attention models presented in the foregoing, a gaze control of an active vision system is tailored to a specific task.

In order to control the gaze of a camera system it is necessary to integrate dynamically different visual cues. For example, different independent depth cues (e.g., image blur, stereo disparity, and motion disparity) have to be combined in a consistent manner: focus-based ranging is most reliable for nearby objects while other cues such as camera motion or stereo disparity can be more useful for greater distances. Furthermore, strategies for interleaving image acquisition and visual processing are investigated depending on the information content of the images and on the computational complexity of a given visual task. Multiscale approaches are discussed in order to achieve information available at different levels of resolution.

9.3.3 Hardware and software concepts

The most interesting property in all active vision systems is their interaction with their environment during sensing. Such systems look in a particular direction depending on the given task, change their focus, move to another location, or reach out and touch something. In order to achieve this functionality, hardware beyond the visual sensor itself is required that can be divided into mechanical and electrical components.

The visual system in human beings and many other animals is partially assisted by other sensory organs. Although the importance of the vestibular information for the vestibulo-ocular responses is well known, its contribution is mostly excluded in the discussion of computer-controlled camera platforms. Neglecting further inertial and other extra-retinal stimuli, the only human visual parameters used are geometrical and optical ones. The mechanical degrees of freedom (DoF, or better axes due to the clear dependence between some of them) of the human head can be categorized as follows:

- mechanical DoF of the eyes (superior-inferior (tilt), lateral-medial (pan), cyclotorsion (about the visual axis));
- mechanical DoF of the neck (tilt, pan, lateral tilt movement); and
- optical DoF of the eyes (change of focal length and iris opening).

The construction of camera platforms in different research labs has been inspired by the described biological findings. This leads to more or less anthropomorphic head-eye systems ranging from camera pan and tilt to complete camera heads (including vergence, neck, etc.) [20, 39, 40, 41, 42]. Often camera systems are connected with robot manipulators and mobile robot platforms and use nonvisual sensors (e.g., tactile and inertial sensors) complementary to visual information [43, 44].

By definition, active vision implies an intimate connection between a controllable imaging system, possibly dynamic environment, and computational processes. These processes have to respond to demands originating from the outside world, while the effectors and the processing must react in timely fashion to the goals of visual or robotic applications. In this sense, the active vision enterprise is one of real-time computation. Hardware to support real-time vision involves at least fast image acquisition (frame grabbers), fast low-level image processing, fast interfaces to gaze control effectors and some interface to higher-level computing.

In addition to the hardware requirements it imposes, the real-time constraint places demands on software at several levels. In Chapter 5, a uniform software design is discussed in order to combine conflicting goals: while, on the one hand, highly run-time efficient code and low-level access to hardware is required, on the other hand, a general, platform-independent implementation is desired. Many practical problems arise when real-time systems are to be controlled with a general-purpose workstation. One solution is to add dedicated processors running nonpreemptive operating systems; another solution is the use of special image-processing hardware (e. g., DataCubes and FPGA image processing (Chapter 2)) and real-time operating systems. The requirements in active vision are to reduce processing latency to the point allowing realistic reaction times to events that occur in the view. Given that robust algorithms can be designed and implemented, multimedia architectures (see Chapter 3) and massively parallel architectures have the possibility of meeting this criterion [21, 45, 46, 47].

9.3.4 Eye-hand coordination

Active visual systems should have certain adaptive capabilities that enable them to map sensory data to grasp or similar motor commands. Thus, they can obtain a higher degree of autonomy for acquiring visual data in unstructured or changing environments and can gain further information about objects by handling them. Furthermore, manipulators can accomplish certain tasks under control of an active vision system (visual guidance). By using the feedback provided by vision an accurate positioning of manipulators is possible. Research in this field is often denoted as *sensorimotor coordination* or *sensorimotor association* [48]. Studies deal with the bidirectional association that exists between sensory perception and motor actions. On the one hand, there is a forward association from motor actions to the resulting sensory perception. Accordingly, the prediction of a robot-arm's position, etc., is desired given specific motor commands (*forward kinematics*). On the other hand, it is desirable to estimate motor commands for a desired sensor position (*inverse kinematics*). Several solutions to such kinematics problems are known that can be divided by the underlying processing rules: computational or learning approaches. The computational approaches referred to here use physical laws for an exact determination of the desired quantities. Often certain assumptions are necessary to reduce the computational complexity, especially in the active vision domain with its real-time requirements, but nevertheless there is no unique solution for redundant "joints" [49, 50]. Thus learning methods employing neural networks are often used. These methods do not require a direct computability because they use other methods known as *learning by example* or *learning by doing* [51, 52]. Neural networks are especially appropriate due to their ability to learn complex nonlinear mappings. Furthermore, their robustness against noise or corrupted input is advantageous.

9.4 Examples for active vision environments

Active vision capabilities have especially entered robotics applications because robots offer a straightforward way to integrate tactile, movement, and grasp facilities in image-processing applications. Thus the coupling between sensoric devices (cameras, sonars, laser rangefinders, encoders, etc.), on the one hand, and actoric devices (motors, grippers, etc.), on the other, can be demonstrated. Nevertheless, one can divide the applications according to the role vision plays. In the robotics domain vision is used for navigation tasks, for example, to detect obstacles in the pathway and to look for landmarks for self-localization. The following compilation of active vision environments is a survey of running research projects and is by no means exhaustive. In the first applications vision and other sensor inputs are used mainly for navigation:

- The U Bonn investigates autonomous robots within the *RHINO*-project [53] (`http://www.informatik.uni-bonn.de/~rhino`). The Rhino robot moves autonomously in office environments applying collision avoidance and path planning mechanisms. It uses artificial neural nets and knowledge data banks about the rooms to build up representations of the scenes. In this way, all algorithms shall work in a real-time software environment. The system uses camera input, sonar echoes and laser range values to analyze the current path and to classify objects. Rhino was utilized as a museum tour guide at the Deutsches Museum Bonn. Within this application, a camera platform was used only to increase the user acceptance.
- At the UBW München, visual guided vehicles have been developed (Chapter 28, [54], and `http://www.unibw-muenchen.de/campus/LRT/LRT13/Fahrzeuge/VaMP-E.html`). The vehicles are able to

drive autonomously by active control of steering, brakes, and accelerator pedal. The whole control system rests upon visual input and is, therefore, equipped with several movable cameras. Thus, multiple images at different zoom levels can be grabbed. In this way, an active, binocular observer that can also track other vehicles is possible. Experiments showed a high robustness and accuracy.

- The institute of robotics at ETH Zürich investigates autonomous mobile robots as interactive and cooperative machines. Within the *MOPS*-project (http://enterprise.ethz.ch/research/mops), an autonomous robot is used for internal mail distribution [55]. The robot is able to navigate very accurately based on the recognition of natural landmarks. Therefore, geometric structures are matched by looking for straight lines, cylinders, corners, etc., within range data. The robot picks up mail boxes and delivers them to the addressee. Thus an interaction with humans is possible via wireless communication links to give orders and to deal with exceptional situations.
- The robotics institute at Carnegie Mellon University is part of the team building a prototype Automated Highway System (AHS) ([56], http://almond.srv.cs.cmu.edu/afs/cs/misc/mosaic/common/omega/Web/Groups/ahs). The goal of the project is to create the specifications for vehicles and roadways that will provide hands-off, feet-off, computer-controlled driving with greatly improved safety and throughput. Cars, trucks, and buses equipped for the AHS will be able to drive alongside non-AHS capable vehicles on the same roadways.

Beyond vision is also used for high-level tasks. Vision is not only regarded here as a tool for navigation but also for tasks such as object localization, identification, or tracking. In the following a few examples are outlined:

- At the TU Berlin the autonomous flight robot *MARVIN* (http://pdv.cs.tu-berlin.de/forschung/TubRob/tubrob97.html) is developed (see [57] for a predecessor project). It is an extended model helicopter, which shall take part in aerial robotics competitions. In addition to some sensors for stabilizing the device, a camera is used to fulfill the desired tasks: find and classify objects.
- Face recognition is coupled with active vision capabilities at the U Bochum [58]. Starting with image acquisition a face candidate segmentation first decides whether a face is present. This is solved by low-level processes looking in parallel for form, motion and color properties of faces. The 2-D position and size measurements are tracked by a steady-state Kalman filter. High-ranked regions that can be tracked over a certain time are checked by a neural net for the presence of a face resulting in a candidate window that is passed to

the face-recognition module. Within the *NEUROS*-project the robot *ARNOLD* was build to investigate the design of a human-like, autonomous service robot ([43] and http://www.neuroinformatik.ruhr-uni-bochum.de/ini/PROJECTS/NEUROS/NEUROS.html). The robot is guided by visual input only, because visual sensors are applicable to a wide range of possible future tasks. The camera head used is equipped with four cameras that provide both an overall view and a binocular foveated view of particular objects. An exemplary application of Arnold consists of finding humans in its environment based on skin color segmentation. After recognizing, it shall point with its gripper onto the hand of the target.

- At the University of Stuttgart, autonomous mobile robots are used for navigation in unknown environments and for the recognition of complex and very similar objects ([59] and http://www.informatik.uni-stuttgart.de/ipvr/bv/comros). Within the research projects, the problem of cooperative image processing is addressed, that is, multiple robots shall investigate an object from different perspectives. Thus, the system behavior of one robot is influenced not only by the visual sensory data, but also by the interaction with its neighbors. Within an exemplary behavior "tracking and recognition of objects," a movement analysis module segments the scene according to motion and delivers a region of interest to the recognition module. The recognition modules of the multiple robots are linked by a so-called contract-net protocol.
- The University of Tübingen also focuses on the problem of cooperative behavior of robots ([60] and http://www-ra.informatik.uni-tuebingen.de/forschung/welcome-e.html). For example, robots should autonomously form a queue. Within other projects the main focus is on the design of autonomous mobile robots [61].
- At the TU Ilmenau the behavioral organization of interactive learning systems is investigated ([44] and http://cortex.informatik.tu-ilmenau.de/forschung.html). The robot *MILVA* is to interact with humans in a direct and natural way. A station scenario has been used as an exemplary environment. The robot is to react on gestures given by passengers and assist them in finding platforms, transport baggage, etc. This scenario involves capabilities such as recognition of hands and baggage, tracking and following of persons, collision avoidance, and navigation.
- At the University of Paderborn, a system for an autonomous disassembly of cars is under development within the *DEMON*-project (see Chapter 8, [62], and http://getwww.uni-paderborn.de/GetLab/GetResearch/GetNeuralVision/getDemon.html). First, car wheels shall be disassembled. The system has to determine the spatial positions of the parts with sufficient accuracy. A robot arm equipped

with specialized tools and controlled by a stereo camera system is to disassemble the cars successively. Objects are learned holistically as a set of characteristic subobjects and later recognized from an arbitrary distance and in an arbitrary orientation. These invariances are accomplished by a normalization based on estimations. The geometric relations of the subobjects shall be combined in a knowledge-based system to efficiently model an entire object.

- At the University of Ulm, the organization of useful interactions between methods of subsymbolic and symbolic information processing, in particular, neural networks and knowledge-based systems, in an autonomous vehicle is investigated ([63] and `http://www.uni-ulm.de/SMART/index.e.html`). The main goal is the development of a learnable cognitive apparatus—a central representation of the senso-motoric situation. This representation forms the basis for the robot's behavior.
- At the University of Hamburg, active vision systems are investigated within the *NAVIS*-project (Chapter 27 and `http://ima-www.informatik.uni-hamburg.de/Research/NAVIS/navis.html`). The system visually explores its environment, locates interesting 2-D and 3-D objects and recognizes them under control of several invariance capabilities. Neural models of visual information processing from psychophysical and neurophysiological results are deduced and used for the development of technical image analysis systems. Current work comprises visual attention and gaze control, 2-D and 3-D object recognition, color image processing, stereo image analysis, and object tracking. A case study of a domino-playing robot is under development [64].
- At Stanford University, autonomous robots are investigated ([65] and `http://underdog.stanford.edu`). The *intelligent observer* is able to follow objects whereas it shall communicate with high-level commands such as "follow next moving object." The project brings together concepts and algorithms from computer vision, motion planning, and computer graphics in order to create a robust, useful, and integrated system. One important aspect of the system is that vision itself, often viewed merely as a mechanism for aiding robot navigation, becomes the central objective of the system. The robot carries one or more cameras that allow it to track objects while at the same time sensing its own location. Five modules were developed: landmark detection; target tracking; motion planning; user interface; and motion control. The first two modules use vision capabilities. For landmark detection, special marks are placed on the ceiling at known positions throughout the robot's workspace. Once a landmark is localized, the positions of the marks inner structures are computed and their intensities are read from the image to de-

termine the 16 binary values they represent. Four of these values are used to disambiguate the landmark's orientation, and the others encode the landmark's unique ID. Target tracking is used by the observer to detect and track a moving target. A target consists of a number of black, vertical bars on a white cylindrical "hat" that sits on top of the target robot. The tracking algorithm detects these bars in each image and, given the camera parameters and the physical size of the bars, computes the target's location relative to the camera.

- At the University of Rochester, wheelchairs were turned into mobile robots ([66] and `http://www.cs.rochester.edu/research/mobile/main.html`) by adding on-board computational power and a few sensors. Among other things, the mobile robot should follow a target placed on the posterior part of another mobile robot platform controlled manually. To reduce complexity, a specially constructed 3-D calibration object is used as the target. To achieve real-time performance, the system is divided in a low-level part to locate quickly the regions of interest in each scene and then to focus its attention exclusively on them. A high-level module carries out geometrical analysis and tracking. The tracking module consists of four parts: prediction; projection; measurement; and backprojection. All steps can greatly be simplified by using *a priori* known target configurations.
- The CVAP laboratory at the Royal Institute of Technology Stockholm investigates a system approach for research in active vision ([67] and `http://www.nada.kth.se/cvap`). Some key issues have been identified: use of multiple types of visual information, attention, and figure-ground segmentation, cues from motion and depth cues from binocular disparities should be given priority. Thus, objects can be segmented for faster additional processing. Attention mechanisms are used to emphasize the effectiveness of motion and 3-D cues.

9.5 Applications for active vision devices

In Section 9.4, some examples for active vision applications available today or intended for the near future are presented. In addition, many more applications seem possible. With growing computational power and the availability of standard sensory systems together with dedicated analysis algorithms, new areas of applications will emerge. In principle, active vision capabilities can be used effectively among others in the followings fields:

- Industrial robotics: Robots for industrial applications used so far lack some kind of intelligent behavior. They are able to repeat the programmed action very accurately, but the main assumption is a constant working environment. Even small changes in the locations of tools cause problems. Here, active vision provides a way to react in an effective way. The long and strenuous reprogramming of robots for new applications can be reduced. On the basis of goal formulations, the robots can autonomously adapt to new environments with new subtasks.
- Service robotics: Within the service domain, a great range of applications suitable for robots is developing. Service robots are mobile devices equipped with adequate sensors to relieve humans of small duties. Starting with simple tasks such as a vision-guided vacuum cleaner or an autonomous post deliverer, such systems will find their place in everyone's life. Especially, the support for handicapped persons demands autonomous systems to offer improved quality of life (see Chapter 26 for a system that enables blind people to experience visual impressions).
- Autonomous vehicles: Active vision systems installed in vehicles provide a way to assist the driver for a safe journey, to increase vehicle throughput, or to autonomously solve tasks by controlling handling devices. Especially, in hazardous environments such systems would find many applications. Active vision methods are the means to solve navigation tasks by handling all stimuli in an adequate time period by concentrating on important subregions first. In conjunction with industrial robots, autonomous vehicles can take on parts of the manufacturing process such as in inventory administration, building industry, etc. In Section 23.3 and Section 28.6, two autonomous systems are introduced.
- Man-machine interaction: Through active vision methods, the interaction between computer and operator can become more user-friendly. In conjunction with voice recognition, the user no longer needs to insert commands by hand. Image processing can resolve ambiguous spoken words. Mouth movements can be interpreted by communicating with tracking facilities that provide the operator's location. A determination of gaze directions by active vision methods also seems possible.
- Surveillance/monitoring: Devices working with active vision can profit manyfold. An active observer can determine unauthorized actions, on the one hand, and is able to patrol certain regions autonomously, on the other hand, if installed on a mobile system. Furthermore, signature verifications systems are imaginable by using tracking mechanisms. Section 22.6 and Section 15.3 outline body tracking systems.

9.6 Conclusion

Active vision as a new paradigm arose from Marr's computational approach of vision. In his book *Vision*, he formed the foundation for a clear, biological inspired, mathematical, symbolic solution for perceptual tasks. His ideas still have great influence on current researchers. Limitations of his approach became obvious as researchers wanted to address complex real-world problems. Often, a straight mathematical mapping of visual information to higher abstraction levels is not possible. Marr's view on vision as a reconstruction process leads to very specialized applications but not to a general visual "problem solver." Mainly, Marr did not take into account the involvement of visual capabilities in a specific visual task of an autonomous observer. By using motoric facilities, many ill-posed problems can become well-posed through the possibility of gaining more visual information from different viewpoints. Therefore, the new view on vision as an active acquisition process developed. Multiple key aspects of active vision such as attention, gaze control, hardware and software concepts, and eye-hand coordination were identified. All these techniques have to be considered as task-dependent under real-time conditions. During recent years, many interesting applications using active vision capabilities have been developed. They have found their way especially into robotics applications, whereas vision is used both for navigation assistance and as a tool for high-level tasks. Besides these, many more future applications seem to be imaginable.

9.7 References

[1] Pentland, A. (ed.), (1986). *From Pixels to Predicates: Recent Advances in Computational and Robot Vision.* Norwood: Ablex.

[2] Bajcsy, R., (1985). Active perception vs. passive perception. In *Proc. 3rd Workshop on Computer Vision: Representation and Control*, pp. 55–59. Bellair.

[3] Bajcsy, R., (1988). Active perception. *Proc. IEEE*, **76**(8):996–1005.

[4] Aloimonos, Y., Weiss, I., and Bandopadhay, A., (1987). Active vision. In *Proc. 1. International Conference on Computer Vision*, pp. 35–54. London.

[5] Marr, D., (1978). Representing visual information—a computational approach. In *Computer Vision Systems*, E. R. Hanson and E. M. Riseman, eds., pp. 61–80. New York: Academic Press.

[6] Marr, D., (1982). *Vision.* New York: Freemann.

[7] Newell, A. and Simon, H. A., (1976). Computer science as empirical enquiry: symbol and search. *Comm. of the ACM*, **19**:113–126.

[8] Marr, D. and Hildreth, E., (1980). Theory of edge detection. In *Proc. Royal Society of London*, number 207 in Series B, pp. 187–217.

[9] Bülthoff, H., Edelman, S., and Tarr, M., (1995). How are three-dimensional objects represented in the brain? *Cerebral Cortex*, **5**(3):247–260.

[10] Perrett, D., Oram, M., Harries, M., Bevan, R., Hietanen, J., Benson, P., and Thomas, S., (1991). Viewer-centered and object-centered coding of heads in the macaque temporal cortex. *Experimental Brain Research*, **86**:159–173.

[11] Tanaka, K., (1996). Inferotemporal cortex and object vision. *Annual Reviews in Neuroscience*, **19**:109–139.

[12] Aloimonos, Y., (1995). Guest editorial: qualitative vision. *International Jour. Computer Vision*, **14**(2):115–117. Special Issue on Qualitative Vision.

[13] Aloimonos, Y. and Shulman, D., (1989). *Integration of Visual Modules. An Extension of the Marr Paradigm*. Boston: Academic Press.

[14] Fermüller, C. and Aloimonos, Y., (1994). *Vision and Action*. Technical Report CAR-TR-772, CS-TR-3305, Computer Vision Laboratory. Center for Automation Research. University of Maryland.

[15] Landy, M. S., Maloney, L. T., and Pavel, M. (eds.), (1996). *Exploratory Vision. The Active Eye*. New York: Springer.

[16] Swain, M. J. and Stricker, M., (1991). *Promising directions in active vision*. Technical Report CS 91-27, University of Chicago.

[17] Viéville, T., (1994). *A few Steps Towards 3D Active Vision*. Technical Report 1, INRIA.

[18] Aloimonos, Y., (1993). Introduction: Active vision revisited. In *Active perception*, Y. Aloimonos, ed. Hillsdale: Lawrence Erlbaum.

[19] Aloimonos, Y., (1994). What I have learned. *CVGIP: Image Understanding*, **60**(1):74–85.

[20] Ballard, D. H., (1991). Animate vision. *Artificial Intelligence*, **48**:57–86.

[21] March, B., Brown, C., LeBlanc, T., Scott, M., Becker, T., Das, P., Karlsson, J., and Quiroz, C., (1992). Operating system support for animate vision. *Jour. Parallel and Distributed Computing*, **15**(2):103–117.

[22] Ballard, D. H. and Brown, C. M., (1992). Principles of animate vision. *CVGIP: Image Understanding*, **56**(1):3–20.

[23] Bajcsy, R., (1996). From active perception to active cooperation—fundamental processes of intelligent behavior. In *Visual Attention and Cognition*, W. H. Zangemeister et al., eds., pp. 309–321. Amsterdam, North Holland: Elsevier Science.

[24] Bajcsy, R. and Campos, M., (1992). Active and exploratory perception. *CVGIP: Image Understanding*, **56**(1):31–40.

[25] Califano, A., Kjeldsen, R., and Bolle, R., (1992). *Active Foveal Vision*. Technical Report RC 18557, IBM.

[26] Reitböck, H. J. and Altmann, J., (1984). A model for size and rotation-invariant pattern processing in the visual system. *Biological Cybernetics*, **51**:113–121.

[27] Schwarz, E. L., (1981). Cortical anatomy, size invariance and spatial frequency analysis. *Perception*, **10**:831–835.

[28] Hartmann, G., Drüe, S., Kräuter, K. . O., and Seidenberg, E., (1993). Simulations with an artificial retina. In *Proc. of the World Congress on Neural Networks*, pp. III–689 – III–694.

[29] Tistarelli, M. and Sandini, G., (1992). Dynamic aspects in active vision. *CVGIP: Image Understanding*, **56**(1):108–129.

[30] Podgorny, P. and Shephard, R. N., (1983). The distribution of visual attention over space. *Jour. Experimental Psychology: Human Perception and Performance*, **9**:380–393.

[31] Posner, M. I., Snyder, C. C. R., and Davidson, B. J., (1980). Attention and the detection of signals. *Jour. Experimental Psychology: General*, **109**: 160–174.

[32] Clark, J. J. and Ferrier, N. J., (1988). Modal control of an attentive vision system. In *IEEE 2nd Int. Joint Conf. on Computer Vision*, pp. 514–523.

[33] Koch, C. and Ullman, S., (1985). Shifts in selective visual attention. Towards the underlying neural circuitry. *Human Neurobiology*, **4**:219–227.

[34] Egner, S., (1998). *Zur Rolle der Aufmerksamkeit für die Objekterkennung. Modellierung, Simulation, Empirie.* PhD thesis, University of Hamburg.

[35] Olshausen, B. A., Anderson, C. H., and Van Essen, D. C., (1994). A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *Jour. Computational Neuroscience*, **2** (1):45–62.

[36] Tsotsos, J. K., (1993). An inhibitory beam for attentional selection. In *Spatial Visions in Humans and Robots*, L. Harris and M. Jenkin, eds., pp. 313–331. Cambridge: Cambridge University Press.

[37] Giefing, G., Janßen, H., and Mallot, H. A., (1992). Saccadic object recognition with an active vision system. In *Proc. Int. Conf. on Pattern Recognition*, pp. 664–667.

[38] Milanese, R., Wechsler, H., Gil, S., Bost, J., and Pun, T., (1994). Integration of bottom-up and top-down cues for visual attention using nonlinear relaxation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 781–785.

[39] Krotkov, E. P., (1989). *Active Computer Vision by Cooperative Focus and Stereo.* New York: Springer.

[40] Pahlavan, K., (1996). Designing an anthromorphic head-eye system. In *Visual Attention and Vision*, W. H. Zangemeister et al., eds. New York: Elsevier.

[41] Trapp, R. and Drüe, S., (1996). Ein flexibles binokulares Sehsystem: Konstruktion und Kalibrierung. In *Proc. Artificial Intelligence*, B. Mertsching, ed., pp. 32–39. Sankt Augustin: Infix.

[42] Viéville, T., Clergue, E., Enriso, R., and Mathieu, H., (1995). Experimenting with 3-D vision on a robotic head. *Robotics and Autonomous Systems*, **14** (1):1–27.

[43] Bergener, T., Bruckhoff, C., Dahm, P., Janßen, H., Joublin, F., and Menzner, R., (1997). Arnold: An anthropomorphic autonomous robot for human environments. In *Proc. of SOAVE '97—Selbstorganisation von Adaptivem Verhalten*, H.-M. Groß, ed., pp. 25–34. Ilmenau: VDI Verlag.

[44] Groß, H.-M. and Böhme, H.-J., (1997). Verhaltensorganisation in interaktiven und lernfähigen Systemen—das MILVA Projekt. In *Proc. of SOAVE '97—Selbstorganisation von Adaptivem Verhalten*, H.-M. Groß, ed., pp. 1–13. Ilmenau: VDI Verlag.

[45] Büker, U. and Mertsching, B., (1995). Parallel evaluation of hierarchical image databases. *Jour. Parallel and Distributed Computing*, **31(2)**:141–152.

[46] Mirmehdi, M. and Ellis, T. J., (1993). Parallel approach to tracking edge segments in dynamic scenes. *Image and Vision Computing*, **11(1)**:35–47.

[47] Pitas, I., (1993). *Parallel Algorithms for Digital Image Processing, Computer Vision and Neural Networks*. New York: John Wiley and Sons.

[48] Versino, C. and Gambarella, L. M., (1995). Learning the visuomotor coordination of a mobile robot by using the invertible Kohonen map. In *Proceedings of the International Workshop on Artificial Neural Networks*, pp. 1084–1091. Berlin.

[49] Kircanski, N., Hui, R., Shimoga, K. B., and Goldenberg, A. A., (1994). Real-time computational aspects of multiple manipulator systems. *Jour. Intelligent and Robotic Systems*, **9(1/2)**:101–120.

[50] Walker, I. D. and Cavallaro, J. R., (1994). Parallel VLSI architectures for real-time kinematics of redundant robots. *Jour. Intelligent and Robotic Systems*, **9(1/2)**:25–43.

[51] Lu, B.-L. and Ito, K., (1995). Regularization of inverse kinematics for redundant manipulators using neural network inversions. In *Proceedings of the IEEE International Conference on Neural Networks, Perth*, pp. 2726–2731.

[52] Torras, C., Cembrano, G., Millan, J. R., and Wells, G., (1995). Neural approaches to robot control: four representative applications. In *From Natural to Artificial Neural Computation, Proceedings of the IWANN '95*, J. Mira and F. Sandoral, eds., pp. 1016–1035. Berlin: Springer.

[53] Buhmann, J., Burgard, W., Cremers, A. B., Fox, D., Hofmann, T., Schneider, F., Strikos, J., and Thrun, S., (1995). The mobile robot rhino. *AI Magazin*, **16(2)**:31–38.

[54] Maurer, M. and Dickmanns, E., (1997). A system architecture for autonomous visual road vehicle guidance. In *IEEE Conf. on Intelligent Transportation Systems*. Boston.

[55] Vestli, S. and Tschichold, N., (1996). MoPS, a system for mail distribution in office-type buildings. *Service Robot: An International Journal*, **2**(2): 29–35.

[56] Bayouth, M. and Thorpe, C., (1996). An AHS concept based on an autonomous vehicle architecture. In *Proc. of the 3rd Annual World Congress on Intelligent Transportation Systems*.

[57] Brandenburg, U. W., Finke, M., Hanisch, D., Musial, M., and Stenzel, R., (1995). TUBROB—an autonomously flying robot. In *Proc. AUVS Symposium*, pp. 627–636. Washington D.C.

[58] Vetter, V., Zielke, T., and von Seelen, W., (1997). Integrating face recognition into security systems. In *Audio- and Video-based Biometric Person*

Authentication, J. Bigün, G. Chollet, and G. Borgefors, eds., Lecture Notes in Computer Science 1206, pp. 439–448. Berlin: Springer.

[59] Oswald, N. and Levi, P., (1997). Cooperative vision in a multi-agent architecture. In *Image Analysis and Processing*, LNCS 1310, pp. 709–716. New York: Springer.

[60] Zell, A., Hirche, J., Jung, D., and Scheer, V., (1997). Erfahrungen mit einer Gruppe mobiler Kleinroboter in einem Roboterpraktikum. In *21. Deutsche Jahrestagung für Künstliche Intelligenz, Freiburg*, Workshop 08 Kognitive Robotik.

[61] Zell, A., Feyrer, S., Ebner, M., Mojaev, A., Schimmel, O., and Langenbacher, K., (1997). Systemarchitektur der autonomen mobilen Roboter Robin und Colin der Universität Tübingen. In *Proc. of SOAVE '97 - Selbstorganisation von Adaptivem Verhalten*, H.-M. Groß, ed., pp. 151–155. Ilmenau: VDI Verlag.

[62] Götze, N., Stemmer, R., Trapp, R., Drüe, S., and Hartmann, G., (1998). Steuerung eines Demontageroboters mit einem aktiven Stereokamerasystem. In *Workshop Dynamische Perzeption*. Bielefeld.

[63] Palm, G. and Kraetzschmar, G., (1997). SFB 527: Integration symbolischer und subsymbolischer Informationsverarbeitung in adaptiven sensomotorischen Systemen. In *Informatik '97—Informatik als Innovationsmotor. Proceedings der 27. Jahrestagung der Gesellschaft für Informatik*, M. Jarke, K. Pasedach, and K. Pohl, eds. Berlin: Springer.

[64] Bollmann, M., Hoischen, R., Jesikiewicz, M., Justkowski, C., and Mertsching, B., (1999). Playing domino: A case study for an active vision system. In *Proc. of the ICVS '99, Las Palmas de Gran Canaria*. (to appear in 1999).

[65] Becker, C., González-Baños, H., Latombe, J. C., and Tomasi, C., (1995). An intelligent observer. In *Fourth International Symposium on Experimental Robotics, ISER'95*. Stanford.

[66] Bayliss, J. D., Brown, C. M., Carceroni, R. L., Eveland, C., Harman, C., Singhal, A., and Van Wie, M., (1997). *Mobile Robotics 1997*. Technical Report 661, Dept. of Computer Science, University of Rochester.

[67] Eklundh, J.-O., Nordlund, P., and Uhlin, T., (1996). Issues in active vision: attention and cue integration/selection. In *Proc. British Machine Vision Conference*, pp. 1–12.

10 The Global Algebraic Frame of the Perception-Action Cycle

Gerald Sommer

Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel, Germany

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

10.1 Introduction

In the last decade the maturation of a new family of intelligent systems has occured. These systems are designed to model the behavior-based paradigm. The roots of the paradigm can be found in the rather disparate disciplines of natural science and philosophy. From a technical point of view they can be summarized as the cybernetic interpretation of living systems on several levels of abstraction and granularity. Although not always recognized, the approach aims at the design of autonomous technical systems.

In this chapter we want to consider the design of behavior-based technical systems in the frame of a general mathematical language, that is, an algebra. This algebra should be powerful enough to integrate different contributing scientific disciplines in a unique scheme and should contribute to overcoming some of the current limitations and shortcomings inherent to those disciplines. The language we are using is Clifford algebra [1] in its geometrically interpreted version as geometric algebra [2]. The use of this algebra results from searching for the natural science principles forming the basis of systematic system design according to the behavior-based paradigm of design. Therefore, we have to consider the motivations of proceeding in this way and the ideas basic to this approach. The outline of the chapter will be as follows. In Section 10.2, we will introduce the basic assumptions of the metaphor of intelligent systems we are using. We will derive its usefulness from engineering requirements with respect to the desired features of technical systems. We will then give some arguments in support of a natural science-based design approach and some already identified key points will be discussed. One of the most important of them will be a kind of mathematical equivalence of the perception and action tasks of a system. Perception will be understood as recognition of regular spatiotemporal patterns of the perceivable environment, whose equivalence classes are verified using action. Conversely, action will be understood as generation of regular spatiotemporal patterns in the reachable environment, whose equivalence classes are verified using perception. From this mutual support of perception and action the need of a mathematical framework like that of geometric algebra will be deduced. This will be done in Section 10.3 together with a description of its basic features for modeling geometric entities and operations on these entities in Euclidean space. We will omit the analysis of manifolds. Instead, in Section 10.4 we will demonstrate the use of this algebraic language with respect to some basic problems related to the analysis of manifolds, their recognition, and transformation. There, we demonstrate the impact of the presented algebraic frame for the theory of multidimensional signals and neural computing.

We will follow a grand tour from engineering to philosophy (Section 10.2), from philosophy to natural science (Section 10.2), from natural science to mathematics (Section 10.3), and finally from mathematics to engineering (Section 10.4) with emphasis on Section 10.3.

10.2 Design of behavior-based systems

10.2.1 From knowledge-based to behavior-based systems

The old dream of human beings, that of designing so-called intelligent machines, seemed possible as a result of the famous 1956 Dartmouth conference, during which such disciplines as "artificial intelligence" and "cognitive science" emerged. Following the zeitgeist, both disciplines were rooted in the *computational theory of mind*, and for a long time they were dominated by the symbol processing paradigm of Newell and Simon [3]. In that paradigm, intelligence is the exclusive achievement of human beings and is coupled with the ability of reasoning on categories.

Because of the assumed metaphorical power of computers to be able to interpret human brain functions, it is not surprising that this view has been inverted, that is, to be put to work to design intelligent machines. The strong relations of the metaphor to symbolic representations of knowledge with respect to the domain of interest not only enforced the development of *knowledge-based* or *expert systems* (KBS) but also considerably determined the directions of development of robotics [4] and computer vision [5]. The corresponding engineering paradigm of the metaphor is called *knowledge-based system* design.

Although the knowledge-based approach to vision and robotics achieved remarkable success in man-made environments, the final result has been less than satisfactory. There is no scaling of the gained solutions to natural environments. As well, within the considered frame, system performance is limited not only from an engineering point of view. This results from the natural limitations of explicit modeling of both the domain and the task. Therefore, the systems lack robustness and adaptability. Yet these properties are the most important features of those technical systems for which engineers strive. The third drawback of the most contemporary available systems is their lack of stability.

All three properties are strongly related to the philosophy of system design. The problems result from the metaphor of intelligent machines. From a cybernetic point of view, all biological systems, whether plants, animals or human beings, are robust, adaptable and stable systems, capable of perceiving and acting in the real world. The observation of the interaction of biological systems with their environment and among themselves enables us to formulate another metaphor. This approach to intelligence is a *socioecological theory of competence*, which we will call *metaphor of biological competence*. Its engineering paradigm is called *behavior-based system* design.

10.2.2 Metaphor of biological competence

Biological competence is the capability of biological systems to do those things right that are of importance for the survival of the species and/or the individual. We use the term *competence* instead of intelligence because the last one is restricted to the ability to reason, which only human beings and some other primates possess. In contrast to *intelligence*, competence is common to all biological systems. It summarizes rather different capabilities of living organisms on several levels of consideration as those of perception, action, endocrine and thermal regulation or, of the capability of engaging in social relations.

For an observer of the system, its competence becomes visible as an adequate activity. The mentioned activity is called *behavior*, a term borrowed from ethology. Of course, behavior can also happen internally as recognition or reasoning. It can be reactive, retarded or planned. In any case, it is the answer of the considered system to a given situation of the environment with respect to the goal or *purpose*, and is both caused by the needs and constrained by the limits of its physical resources. The attribute adequate means that the activated behavior contributes something to reach the goal while considering all circumstances in the right way. This is an expression of competence.

Behavior obviously subsumes conceptually at least three constituents: an *afferent* (sensoric) and an *efferent* (actoric) interaction of the system with its *environment*, and any *kind of relating the one* with the other (e. g. by mapping, processing, etc.). Because of the mutual support and mutual evaluation of both kinds of interaction a useful system theoretical model is cyclic arrangement instead of the usual linear input/output model. If the frame of system consideration is given by the whole biological system (a fly, a tree or a man), we call this arrangement the *perception-action cycle* (PAC). Of course, the mentioned conceptual separation does not exist for real systems. Therefore, having artificial systems follow the behavior-based system (BBS) design of the metaphor of biological competence is a hard task. Koenderink [6] called *perception* and *action* two sides of the same entity.

An important difference between behavior and PAC thus becomes obvious. While behavior is the observable manifestation of competence, the perception-action cycle is the frame of dynamics the system has to organize on the basis of the gained competence. This distinction will become important in Section 10.2.4 with respect to the designers' task.

Competence is the ability of the system to organize the PAC in such a way that the inner degrees of freedom are ordered as a result of the perceived order of the environment and the gained order of actions with respect to the environment. A competent system has the capability to separate relevant from irrelevant percepts and can manage its task in the total complexity of concrete environmental situations. This is what is called *situatedness of behavior*. With the term *corporeality of behavior* the dependence of the competence from the specific needs and limitations of the physical resources should be named. This also includes determining the influence of the environment on the adaptation of the corporeality during phylogenesis. Both features of behavior are in sharp contrast to the computational theory of mind. Gaining competence is a double-track process. On the one hand, learning from experience is very important. But learning from scratch the entirety of competence will be too complex with respect to the individual lifespan. Therefore, learning has to be biased by either acquired knowledge from phylogenesis or from the learner.

In the language of *nonlinear dynamic systems*, behavior has the properties of an *attractor*. This means that it is *robust* with respect to small distortions and *adaptable* with respect to larger ones. Because of its strong relation to purpose, behavior has the additional properties of usefulness, efficiency, effectiveness, and suitability. From this it follows that an actual system needs a plethora of different behaviors and some kind of control to use the right one. This control may be, for example, event-triggering or purposive decision-making using either intuition or reasoning, respectively. Behaviors can be used to conceptualize a system as a conglomerate of cooperating and competing perception-action cycles. The loss of one behavior does not result in the breakdown of the whole system. This is the third important property from the engineer's point of view, and is called *stability*.

10.2.3 From natural to artificial behavior-based systems

A general consequence of the sketched metaphor of biological competence seems to be that each species and even each individual needs its own architecture of behaviors. This is true to a great extent. This consequence results from different physical resources and different situative embeddings of the systems. For instance, the vision task is quite different for an ant, a frog, or a chimpanzee. There is no unique general theory of biological competences as regards vision; this is in contrast to the postulation by Marr [5]. This conclusion may result either in resignation or in dedicated design of highly specialized systems. Because of the limitations of knowledge-based approaches the last way is the one that contemporary engineers are adopting for industrial applications. However, there is no reason to follow it in the longer term.

Indeed, biological competence cannot be formulated as laws of nature in the same way as has been the case for laws of gravity. We should bear in mind that competence takes these natural laws into account. There must be several more or less general principles that biological systems use towards learning or applying competence. What at a first glance is recognized as a set of heuristics will be ordered with respect to its common roots. While as a matter of basic research these principles have to be identified, over the long term it is hoped that we will be able to follow the engineering approach. This need of identification requires observation of natural systems using, for example, ethology, psychophysics and neural sciences, and the redefining of already known classes of problems.

From a system theoretical point of view, behavior-based systems are *autonomous systems*. They are *open systems* with respect to their environment, *nonlinear systems* with respect to the situation dependence of their responses to sensory stimuli, and *dynamic systems* with respect to their ability to activate different behaviors. In traditional engineering, the aggregation of a system with its environment would result in a *closed system*. As the mutual influences between the environment and the original system are finite ones with respect to a certain behavior, this kind of conception seems to make sense. We call this approach the *top-down design* principle. The observation of an interesting behavior leads to modeling and finally to the construction and/or implementation of the desired function. However, this mechanistic view of behavior has to be assigned to the computational theory of mind with all the mentioned problems. Instead, we indeed have to consider the system as an open one.

10.2.4 Bottom-up approach of design

In order to make sure that the behavior will gain attractor properties, the designer has to be concerned that the system can find the attractor basin by self-organization. The system has to sample stochastically the space of its relations to the environment, thus to find out the regularities of perception and action and to respond to these by self-tuning of its parameters. This is *learning by experience* and the design principle is bottom-up directed.

The *bottom-up design* of behavior-based systems requires the designer to invert the description perspective with respect to behavior to the synthesizing perspective with respect to the perception-action cycle. Instead of programming a behavior, *learning* of competence has to be organized. The resulting behavior can act in the world instead of merely a model of it. Yet correctness that can be proved will be replaced by observable success.

A pure bottom-up strategy of design will make no sense. We know that natural systems extensively use knowledge or *biasing* as genetic information, instincts, or in some other construct. Both learning paradigms and the necessary quantity and quality of biasing are actual fields of research in neural computation. For instance, *reinforcement learning* of sonar-based navigation of a mobile robot can be accelerated by a factor of a thousand by partitioning the whole task into situation dependent subtasks and training them separately. Another approach with comparable effect will be gained by delegation of some basic behaviors to programmed instincts. The division between *preattentive visual tasks* and *attentive visual tasks* or the preprocessing of visual data using operations comparable to simple cells in the primary visual cortex plays a comparable role in the learning of visual recognition.

Thus, to follow the behavior-based paradigm of system design requires finding good learning strategies not only from *ontogenesis* of biological systems but also from *phylogenesis*. The latter one can be considered as responsible for guaranteeing good beginning conditions for individual learning of competence. Without the need of explicit representation such kind of knowledge should be used by the designer of artificial behavior-based systems.

The bottom-up design principle is not only related to the learning of regularities from seemingly unrelated phenomena of the environment. It should also be interpreted with respect to the mutual dependence of the tasks. *Oculomotor behaviors* like visual *attention*, *foveation* and *tracking* are basic ones with respect to *visual behaviors* as *estimation of spatial depth* and this again may be basic with respect to *visual navigation*. Because of the forementioned reduction of learning complexity by partitioning a certain task, a bootstrap strategy of design will be the preferential one. Such stepwise extension of competences will not result in quantitatively linear *scaling of capabilities*. But from the interacting and competing competences qualitatively new competences can *emerge*, which are more than the union of the basic ones. From this nonlinear scaling of capabilities it follows that the design of behavior-based systems cannot be a completely determined process.

10.2.5 Grounding of meaning of equivalence classes

In KBS design we know the problem of the missing *grounding of meaning of categories*. In BBS design this problem becomes obsolete as the so-called *signal-symbol gap* vanishes. *Categories* belong to the metaphor of the computational theory of mind. They are only necessary with respect to modeling, reasoning, and using language for communication. Most biological systems organize their life on a precategorical level using *equivalence classes* with respect to the environment and the self. Equivalence classes are those entities that stand for the gained inner order as a result of learning. Their meaning is therefore grounded in the experience made while trying to follow the purpose. In contrast to classical definitions of equivalence classes they have to be multiply supported. This is an immediate consequence of the intertwined structure of the perception-action cycle. If we distinguish between equivalence classes for perception and action, we mean that they are useful with respect to these tasks [7]. This does not mean that they are trained either by perception or action in separation. Instead, their development out of random events is always rooted in both multiple sensory clues using *actoric verification*. Actoric verification means using actions to get useful sensory data. This mutual support of equivalence classes by perception and action has to be considered in the design of special systems for computer vision or visual robotics [8]. A visual navigating system does not need to represent the equivalence classes of all objects the system may meet on its way but useful relations to any objects as equivalence classes of sensorimotorics. We will come back to this point in Sections 10.2.6 and 10.3.2. The grounding of meaning in physical experience corresponds to the construction or selection of the *semantic aspect* of *information*. The *pragmatic aspect* of information is strongly related to the purpose of activity. The *syntactic aspect* of the information is based on sensorics and perception. Cutting of the perception-action cycle into perception and action as separated activities thus will result in the loss of one or another aspect of the forementioned trinity.

10.2.6 General principles of behavior-based system design

We have identified so far the two most important principles of BBS design: the knowledge biased bottom-up approach and the multiple support of equivalence classes. Here we will proceed to identify some additional general principles of design from which we will draw some conclusions for the adequate algebraic embedding of the PAC.

Minimal effort and *purposive opportunism* are two additionally important features of behavior. These result from the limited resources of the system, the inherent real-time problem of the PAC, and the situative embedding of purposively driven behavior. From these two features we can formulate the following required *properties of behavior*:

1. **fast**: with respect to the time scale of the PAC in relation to that of the events in the environment;
2. **flexible**: with respect to changes of the environment in relation to the purpose;
3. **complete**: with respect to the minimal effort to be purposive;
4. **unambiguous**: with respect to the internalized purpose; and
5. **selective**: with respect to purposive activation of alternative behaviors.

These need result in challenging functional architectures of behavior-based systems. The realization of useful strategies has to complete more traditional schemes as pattern recognition or control of motion.

Thus, the cyclic scheme of the PAC will project also to the internal processes of purposive information control. This has been shown by the author [9] with respect to the design of a visual architecture.

As a further consequence there is a need of equivalence classes of *graduated completeness*. It may be necessary and sufficient to use only rough versions of equivalence classes for motion or recognition. This requires their definition as *approximations* and their use in a scheme of *progression*. Ambiguity need not be resolved at all steps but has to vanish at the very end of processing. Besides, if there are other useful ways or hints that result in unambiguous behaviors the importance of a single one will be relative. Simultaneously the *principle of consensus* [10] should be used. Such representations of equivalence classes have to be sliced within the abstraction levels of equivalence classes while traditional schemes only support slicing between abstraction levels. The basis functions of steerable filter design [11] in early vision represent such a useful principle. Another example would be the eigenvalue decomposition of patterns [12] and motion [13].

This quantitative scaling of completeness has to be supplemented by a scheme of *qualitative completeness* of equivalence classes. The most intuitive example for this is given by the task of visual navigation [14]. In this respect only in certain situations will it be necessary to compute a Euclidean reconstruction of 3-D objects. In standard situations there is no need to recognize objects. Rather, holding some dynamically stable relations to conjectured obstacles will be the dominant task. For this task of minimal effort a so-called scaled relative nearness [15] may be sufficient, which represents depth order as qualitative measure for a gaze fixating and moving uncalibrated stereo camera [16].

Here the oculomotor behavior of *gaze fixation* is essential for getting depth order. Looking at a point at infinity does not supply such a clue. The general frame of switching between *metric*, *affine* or *projective representations of space* was proposed by Faugeras [17].

Generally speaking, in behavior-based systems recognition will dominate reconstruction. But in highly trained behaviors even recognition can be omitted. Instead, blind execution of actions will be supported by the multiple-based equivalence classes. This means that, for example, visually tuned motor actions will not need visual feedback in the trained state. In that state the sequence of motor signals permits the execution of motions in a well-known environment using the principle of trust with respect to the stability of the conditions.

There are other diverse general principles that are used by natural systems and should be used for BBS design but must be omitted here due to limited space.

10.3 Algebraic frames of higher-order entities

10.3.1 Perception, action and geometry

In this section we want to examine some essential features of perception and action to conclude hereof a mathematical framework for their embedding in the next sections. In Section 10.2.5 we learned from the tight coupling of both behavioral entities the need for common support of equivalence classes. Here we want to emphasize the *equivalence of perception and action* with respect to their aims from a geometrical point of view. In this respect we implicitly mean visual perception. Mastering geometry indeed is an essential part of both perception and action, see [18] or the following citations:

Koenderink [19]: "The brain is a geometry engine."
Pellionisz and Llinas [20]: "The brain is for (geometrical) representation of the external world."
von der Malsburg [21]: "The capabilities of our visual system can only be explained on the basis of its power of creating representations for objects that somehow are isomorphic to their real structure (whatever this is)."

The *representation problem*, as indicated by the citations, will be of central importance in this section. A system, embedded in Euclidean space and time, is seeking to perceive structures or patterns of high regularity, and is seeking to draw such patterns by egomotion. The equivalence classes of a competent system correspond to smooth manifolds of low local *intrinsic dimension* [12]. While some patterns of motion are globally 1-D structures (as gestures), others (such as facial expressions) are globally 2-D structures but often can be locally interpreted as 1-D structures.

Both perceptual and motor generated patterns are of high *symmetry*. In at least one dimension they represent a certain conception of *invariance* as long as no event (in case of action) or no boundary (in case of perception) causes the need to switch to another principle of invariance. On the other hand, completely uncorrelated patterns have an infinite or at least large intrinsic dimension. The smooth manifolds of the trained state emerge from initially uncorrelated data as a result of learning.

We introduced the equivalence classes as manifolds to correspond with the bottom-up approach of statistical analysis of data from $\mathbb{R}^N$, as is the usual way of neural computation. Within such an approach the equivalence classes can be interpreted as (curved) subspaces $\mathbb{R}^M$, $M < N$, which constitute point aggregations of $\mathbb{R}^N$. This approach is useful from a computational point of view and indeed the paradigm of artificial neural nets has been proven to learn such manifolds in a topology preserving manner (see, e.g., Bruske and Sommer [22]).

Yet such a *distributed representation* of equivalence classes is inefficient with respect to operations, as comparison, decision finding, or planning. Therefore, equivalence classes have to be represented additionally as *compact geometric entities* on a higher level of abstraction. For instance, searching for a chair necessitates representing a chair itself. Although we do not know the best way of representation yet, in the following sections we will develop an approach to represent so-called higher order geometric entities as directed numbers of an algebra. In the language of neural computation these may be understood as the activation of a grandmother neuron. In Section 10.4.2 we will discuss such representations in an algebraically extended multilayer perceptron.

10.3.2 Behavioral modulation of geometric percepts

Animals have to care for globally spatiotemporal phenomena in their environment. They use as behavior-based systems both gaze control and oculomotor behaviors to reduce the complexity of both vision and action [7, 8]. In this context, vision and action are related in two complementary senses:

1. **vision for action** : similar visual patterns cause similar motor actions; and
2. **action for vision** : similar motor actions cause similar visual patterns.

This implicitly expresses the need to understand vision as a serial process in the same manner as action. Global percepts result from a quasi-continuous sequence of local percepts. These are the result of *attentive vision*, which necessarily uses fixation as an oculomotor behavior. Fixation isolates a point in space whose meaning with respect to its environment will be evaluated [23]. Although signal theory supplies a rich toolbox, a complete local structure representation can only be provided by the algebraic embedding of the task, which will be presented in Section 10.4.1.

Tracking as another *gaze control* behavior is useful to transform temporal patterns of moving objects into a stationary state. Conversely a dynamic perception of nonmoving objects results from egomotion of the head. The control of the eyes' gaze direction, especially with respect to the intrinsic time scale of the PAC, is a demanding task if the head and body are able to move. Kinematics using *higher-order entities*, see Section 10.3.9, will also be important with respect to fast navigation in the presence of obstacles. It will be necessary to plan egomotion on the basis of visual percepts while considering the body as a constraint. That problem of trajectory planning is well known as the "piano mover problem." This means that not only distances between points but also between any entities such as *points*, *lines* or *planes* are of importance. In this respect, an alternative to the forementioned sequential process of attentive vision will gain importance. Indeed, attentive vision is accompanied by a fast parallel recognition scheme called preattentive vision. In *preattentive vision* the percepts of internalized concepts of certain regular patterns (also higher-order entities) pop out and cause a shift of attention. All visual percepts origin from retinal sensory stimuli, which are projections from the 3-D environment and have to be interpreted accordingly. The proposed algebraic frame will turn out to be intuitively useful with respect to projective reconstruction and recognition. As outlined in Section 10.2.6, oculomotor behaviors are used for fast switching between the projective, affine or similarity group of the stratified Euclidean space. In section Section 10.3.9 we will show how to support these group actions and those of rigid movements just by switching the signature of the algebra.

10.3.3 Roots of geometric algebra

We have argued in Section 10.3.1 from the position of minimal effort that to hold a PAC running one needs higher-order geometric entities. Besides, we indicated the problems of locally recognizing multidimensional signals, and the problems of locally organizing chained movements in space, respectively. All three problems have a common root in the limited representation capability of linear vector spaces, respectively, of linear *vector algebra*. Although there are available in engineering and physics more powerful algebraic languages, such as Ricci's *tensor algebra* [24] or Grassmann's *exterior algebra* [25], their use is still limited in the disciplines contributing to the design of artificial PAC (see for instance [26, 27]) for the application of tensor calculus for multi-dimensional image interpretation, or [28] with respect to applications of exterior calculus to computer vision. We decided on *geometric algebra* or GA [2, 29] because of its universal nature and its intuitive geometric interpretation. The main bulk of this algebra rests on *Clifford algebra* [1] (see [30] for a modern introduction). Yet it covers also vector algebra, tensor algebra, *spinor algebra*, and *Lie algebra*. In a natural way it includes the algebra of *complex numbers* [31] and that of *quaternionic numbers* (or *quaternions*) [32]. The useful aspect of its universality is its capability of supporting quantitative, qualitative and operational aspects in a homogeneous frame. Therefore, all disciplines contributing to PAC, for example, *computer vision*, *multidimensional signal theory*, *robotics*, and *neural computing* can be treated in one algebraic language. Of course, the existing isomorphism between the algebras enables the decision for one or another. But often such decisions are costly. Such costs include redundancy, coordinate dependence, nonlinearity, or incompatible system architecture. The GA is most commonly used in physics, although some related conceptions as screw theory of kinematics can also be found in robotics [33, 34]. The first attempts on the study of behavior control were published by Tweed et al. [35] and Hestenes [36, 37].

10.3.4 From vector spaces to multivector spaces

A *geometric algebra* G_n is a linear space of dimension 2^n, which results from a *vector space* V_n of dimension n by endowing it with a new type of product called geometric product. The entities $\mathbf{A}$ of G_n are called *multivectors*

$$\mathbf{A} = \sum_{k=0}^{n} \mathbf{A}_k \tag{10.1}$$

where $\mathbf{A}_k = \langle \mathbf{A} \rangle_k$, $k \le n$ are *homogeneous multivectors* of *grade* k or simply k*-vectors*. The GA is a linear and associative algebra with respect to addition and multiplication, endowed with additive and multiplicative identities, algebraically closed and commutative with respect to addition but not with respect to multiplication [2]. Instead, for any $\mathbf{A}, \mathbf{B} \in G_n$ the product $\mathbf{AB}$ is rather complicated related to $\mathbf{BA}$. The product of any two homogeneous multivectors $\mathbf{A}_r$, $\mathbf{B}_s$, $r + s \le n$, results in an *inhomogeneous multivector* $\boldsymbol{C} \in G_n$ consisting of a set of different grade homogeneous multivectors.

$$\mathbf{C} = \mathbf{A}_r\mathbf{B}_s = \langle \mathbf{A}_r\mathbf{B}_s \rangle_{|r-s|} + \langle \mathbf{A}_r\mathbf{B}_s \rangle_{|r-s|+2} + \cdots + \langle \mathbf{A}_r\mathbf{B}_s \rangle_{r+s} \tag{10.2}$$

This equation offers the desired property of the algebra that the product of two entities results in a set of other *entities* of different grade. This can be best understood if we consider the lowest grades k-vectors. They correspond to the following nomenclature: $k = 0:$ *scalars*, $k = 1:$ *vectors*, $k = 2:$ *bivectors*, ...

We start with vectors $\boldsymbol{a}$, $\boldsymbol{b} \in V_n$. They will remain the same in G_n because for any vector $\boldsymbol{a}$ we will have $\mathbf{A}_1 = \langle \mathbf{A} \rangle_1 = \boldsymbol{a}$. Therefore, we will symbolize also in GA a 1-vector as a vector of V_n. Vectors follow the *geometric product*

$$\mathbf{C} = \boldsymbol{ab} = \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \wedge \boldsymbol{b} \tag{10.3}$$

with the *inner product*

$$\mathbf{C}_0 = \langle \boldsymbol{ab} \rangle_0 = \boldsymbol{a} \cdot \boldsymbol{b} = \frac{1}{2}(\boldsymbol{ab} + \boldsymbol{ba}) \tag{10.4}$$

and the *outer product*

$$\mathbf{C}_2 = \langle \boldsymbol{ab} \rangle_2 = \boldsymbol{a} \wedge \boldsymbol{b} = \frac{1}{2}(\boldsymbol{ab} - \boldsymbol{ba}) \tag{10.5}$$

so that $\mathbf{C}$ is a mixture of a scalar and a bivector. Therefore, in Eq. (10.2) the term $\langle \mathbf{A}_r \mathbf{B}_s \rangle_{|r-s|} = \mathbf{A}_r \cdot \mathbf{B}_s$ stands for a pure inner product and the term $\langle \mathbf{A}_r B_s \rangle_{r+s} = \mathbf{A}_r \wedge \mathbf{B}_s$ stands for a pure outer product component. If in Eq. (10.3) $\boldsymbol{a} \wedge \boldsymbol{b} = 0$, then

$$\boldsymbol{ab} = \boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{b} \cdot \boldsymbol{a} = \boldsymbol{ba} \tag{10.6}$$

and, otherwise if $\boldsymbol{a} \cdot \boldsymbol{b} = 0$, then

$$\boldsymbol{ab} = \boldsymbol{a} \wedge \boldsymbol{b} = -\boldsymbol{b} \wedge \boldsymbol{a} = -\boldsymbol{ba} \tag{10.7}$$

Hence, from Eq. (10.6) and Eq. (10.7) follows that *collinearity* of vectors results in commutativity and *orthogonality* of vectors results in anti-commutativity of their geometric product. These properties are also valid for all multivectors.

The expansion of a GA G_n for a given vector space V_n offers a rich hierarchy of structures, which are related to the subspace conception of the algebra. From the vector space V_n that is spanned by n linear independent vectors $\boldsymbol{a}_i = a_i \boldsymbol{e}_i$, $\boldsymbol{e}_i$ unit basis vector, results one unique maximum grade multivector $\mathbf{A}_n = \langle \mathbf{A} \rangle_n$ that factorizes according

$$\mathbf{A}_n = \prod_{k=1}^{n} \boldsymbol{a}_k \tag{10.8}$$

On each other grade k any k linear independent vectors $\boldsymbol{a}_{i_1}, \ldots, \boldsymbol{a}_{i_k}$ will factorize a special homogeneous k-vector, which is called *k-blade* $\mathbf{B}_k$ thus

$$\boldsymbol{B}_k = \prod_{j=1}^{k} \boldsymbol{a}_j \tag{10.9}$$

There are $l = \binom{n}{k}$ linear independent k-blades $\mathbf{B}_{k_1}, \ldots, \mathbf{B}_{k_l}$ that span the linear subspace G_k of all k-vectors $\mathbf{A}_k \in G_n$, so that all $\mathbf{A}_k$, $k = 1, \ldots, n$, with

$$\mathbf{A}_k = \sum_{j=1}^{l} \mathbf{B}_{k_j} \tag{10.10}$$

finally complete with any $\mathbf{A}_0$ the inhomogeneous multivector $\mathbf{A}$ of the algebra $G_n(\mathbf{A})$, following Eq. (10.1). Hence, each k-blade $\mathbf{B}_k$ corresponds to a vector subspace $V_k = \langle G_n(\mathbf{B}_k) \rangle_1$, consisting of all vectors $\boldsymbol{a} \in V_n$ that meet the collinearity condition

$$\boldsymbol{a} \wedge \mathbf{B}_k = 0 \tag{10.11}$$

From this it follows that, as with the vector $\boldsymbol{a}$ and the k-blades $\mathbf{B}_k$, respectively, the vector subspaces $V_k = \langle G_n(\mathbf{B}_k)\rangle_1$ have a unique direction. Therefore, k-vectors are also called *directed numbers*. The direction of $\mathbf{B}_k$ is a *unit k-blade*

$$\mathbf{I}_k \ = \ \boldsymbol{e}_{i_1} \wedge \boldsymbol{e}_{i_2} \wedge \cdots \wedge \boldsymbol{e}_{i_k} \tag{10.12}$$

so that $\mathbf{B}_k = \lambda_k \mathbf{I}_k$, $\lambda_k \in \mathbb{R}$, respectively $\lambda_k \in \mathbf{A}_0$ and $\mathbf{e}_{i_j} \in \langle G_n(\mathbf{I}_k)\rangle_1$. Of course, the same factorization as Eq. (10.9) is valid for $k = n$. The only n-blade $P \equiv \mathbf{B}_n$ is called *pseudoscalar*. Its direction is given by the *unit pseudoscalar* I that squares to

$$I^2 \ = \ \pm 1 \tag{10.13}$$

Finally, a remarkable property of any G_n should be considered. Each subset of all even grade multivectors constitutes an even *subalgebra* G_n^+ of G_n so that the linear space spanned by G_n can be expressed as the sum of two other linear spaces

$$G_n \ = \ G_n^- \ + \ G_n^+ \tag{10.14}$$

Because G_n^- is not closed with respect to multiplication, it does not represent a subalgebra.

10.3.5 Properties of multivector spaces

In this subsection we want to present some basic properties of a GA as a representation frame that demonstrate both its superiority in comparison to vector algebra and its compatibility with some other algebraic extensions.

A multivector $\mathbf{A}$ has not only a direction but also a *magnitude* $|\mathbf{A}|$, defined by

$$|\mathbf{A}|^2 = \langle \tilde{\mathbf{A}}\mathbf{A}\rangle_0 \geq 0 \tag{10.15}$$

where $|A| = 0$ only in case of $A = 0$ and $\tilde{\mathbf{A}}$ is the *reversed* version of $\mathbf{A}$, which results from the reversed ordering of the vectorial factors of the blades. Because these behave as

$$|\mathbf{B}_k|^2 = |\boldsymbol{a}_1 \cdots \boldsymbol{a}_k|^2 = |\boldsymbol{a}_1|^2 \cdots |\boldsymbol{a}_k|^2 \geq 0 \tag{10.16}$$

we get

$$|\mathbf{A}|^2 = |\langle \mathbf{A}\rangle_0|^2 + |\langle \mathbf{A}\rangle_1|^2 + \cdots + |\langle \mathbf{A}\rangle_n|^2 \tag{10.17}$$

The magnitude has the properties of a norm of $G_n(\mathbf{A})$ yet has to be specified, if necessary, by endowing the algebra with a signature, see for example, Section 10.3.8. A geometrical interpretation can be associated with a k-blade. It corresponds with a directed k-dimensional hypervolume of $G_k(\mathbf{A})$, whose magnitude is given by Eq. (10.15) and whose k-dimensional direction is given by Eq. (10.12). The inner product of multivectors expresses a degree of similarity, while the outer product expresses a degree of dissimilarity. The great power of the geometric product results from simultaneously holding both. In other words, the closeness of the GA with respect to the geometric product results from the properties of the pseudoscalar P that uniquely determines the properties of the vector space V_n. Because the unit pseudoscalar I factorizes with respect to a chosen unit k-blade $\mathbf{I}_k$ such that

$$\mathbf{I}_k \mathbf{I}_{n-k} = I \tag{10.18}$$

any G_n relates two mutual orthogonal vector subspaces $V_k = G_1(\mathbf{I}_k)$ and $V_{n-k} = G_1(\mathbf{I}_{n-k})$. This results in the definition of a *dual* k*-vector*

$$\mathbf{A}^*_{n-k} = \mathbf{A}_k I^{-1} = \mathbf{A}_k \cdot I^{-1} \tag{10.19}$$

because $II^{-1} = 1$. Thus, the *duality operation* [38] changes the multivector basis and enables us to consider any entity from its dual aspect. For given $\mathbf{A}_r$ and $\mathbf{B}_s$ the duality of their inner and outer products becomes obvious

$$\mathbf{A}_r \cdot \mathbf{B}^*_s = (\mathbf{A}_r \wedge \mathbf{B}_s)^* \tag{10.20}$$

Because in case of $r + s = n$ it follows

$$P = \lambda I = \mathbf{A}_r \wedge \mathbf{B}_s \tag{10.21}$$

also the scalar λ can be interpreted as the dual of the pseudoscalar P,

$$\lambda = [P] = PI^{-1} = (\mathbf{A}_r \wedge \mathbf{B}_s)I^{-1} = \mathbf{A}_r \cdot \mathbf{B}^*_s \tag{10.22}$$

In Grassmann algebra [25, 28] the extra operator *bracket* [.], is introduced to define exteriors (or extensors) as the subspaces of the algebra. While exteriors are in one-to-one relation to k-vectors, and while Eq. (10.9) corresponds to Grassmann's *progressive product*, the bracket replaces the missing inner product in Grassmann algebra (see Eq. (10.22)) but is not an operation of the algebra. Furthermore, because

$$[P] = [\boldsymbol{a}_1 \boldsymbol{a}_2 \ldots \boldsymbol{a}_n] = \det(V_n) \tag{10.23}$$

the scalar λ may be considered as a definition of the *determinant* of the vector coordinates $\{a_i,\ i = 1, \ldots, n\}$ in V_n. In contrast to this coordinate dependence of the determinant, λ may be computed completely coordinate independent from multivectors as higher-order entities.

In Section 10.3.8 we will show that k-vectors correspond to *tensors of rank k*. If the values of tensors are inhomogeneous multivectors, conventional tensor analysis can be considerably enriched within GA (see e.g., Hestenes and Sobczyk [2]).

10.3.6 Functions, linear transformations and mappings

The aim of this subsection is to introduce some fundamental aspects of GA that are of immediate importance in the frame of PAC design. The first one is the *exponential function* of a multivector $\mathbf{A} \in G_n$ as a mapping $\exp(\mathbf{A}) : G_n \longrightarrow G_n$, algebraically defined by

$$\exp(\mathbf{A}) \equiv \sum_{i=0}^{\infty} \frac{\mathbf{A}^i}{i!} \tag{10.24}$$

The known relation

$$\exp(\mathbf{A}) \exp(\mathbf{B}) = \exp(\mathbf{A} + \mathbf{B}) = \exp(\mathbf{C}) \tag{10.25}$$

is only valid in case of collinearity of $\mathbf{A}$ and $\mathbf{B}$, else $\mathbf{C} \neq \mathbf{A} + \mathbf{B}$. The invalidity of Eq. (10.25) in the algebraic embedding of the N-D Fourier transform will be surmounted in Section 10.4.1. Equation (10.24) can be expressed by the even and odd parts of the exponential series

$$\exp(\mathbf{A}) = \cosh(\mathbf{A}) + \sinh(\mathbf{A}) \tag{10.26}$$

Alternatively, if I is a unit pseudoscalar with $I^2 = -1$ and, if $\mathbf{A}$ is another multivector which is collinear to I, then

$$\exp(\mathbf{IA}) = \cos(\mathbf{A}) + I \ \sin(\mathbf{A}) \tag{10.27}$$

Because of Eq. (10.19) or, equivalently $\mathbf{A}_k I = \mathbf{A}^*_{n-k}$, Eq. (10.27) can be read with respect to a dual multivector exponential.

a) $k = 3 : \mathbf{A}_3 = \mathbf{A}_0 I,\ \mathbf{A}_0 \in \langle G_3(\mathbf{A})\rangle_0$
$$\exp(\mathbf{A}_3) = \cos(\mathbf{A}_0) + I \sin(\mathbf{A}_0)$$
b) $k = 2 : \mathbf{A}_2 = \mathbf{A}_1 I,\ \mathbf{A}_1 = \mathbf{A}_0 \boldsymbol{e}, \mathbf{A}_0 = |\mathbf{A}_1|$
$$\exp(\mathbf{A}_2) = \cos(\mathbf{A}_1) + I \sin(\mathbf{A}_1)$$
$$\exp(\boldsymbol{A}_2) = \cos(\boldsymbol{A}_0) + I\boldsymbol{e} \sin(\boldsymbol{A}_0)$$
c) $k = 1 : \mathbf{A}_1 = \mathbf{A}_2 I,\ \mathbf{A}_1 = \mathbf{A}_0 \boldsymbol{e},\ \mathbf{A}_0 = |\mathbf{A}_1|$
$$\exp(\mathbf{A}_1) = \cosh(\mathbf{A}_0) + \boldsymbol{e} \sinh(\mathbf{A}_0)$$
d) $k = 0 : \mathbf{A}_0 = \mathbf{A}_3 I$
$$\exp(\mathbf{A}_0) = \cosh(\mathbf{A}_0) + \sinh(\mathbf{A}_0)$$

The second point to be considered is the behavior of *linear transformations* $\boldsymbol{L}$ on a vector space V_n with respect to the algebraic embedding into G_n. The extended version $\mathcal{L}$ of the linear transformation $\boldsymbol{L}$ is yet a linear one in G_n and distributive with respect to the outer product.

$$\mathcal{L}(\mathbf{A} \wedge \mathbf{B}) = (\mathcal{L}\mathbf{A}) \wedge (\mathcal{L}\mathbf{B}) \tag{10.28}$$

Because of the preserving property with respect to the outer product, this behavior of a linear transformation is called *outermorphism* [39].

The inner product will not be generally preserved. For any k-blade $\mathbf{B}_k$ we have

$$\mathcal{L}\mathbf{B}_k = (\mathcal{L}\boldsymbol{a}_1) \wedge (\mathcal{L}\boldsymbol{a}_2) \wedge \ldots \wedge (\mathcal{L}\boldsymbol{a}_k) \tag{10.29}$$

and for any multivector $\mathbf{A}$ it will be grade-preserving, thus

$$\mathcal{L}(\langle \mathbf{A} \rangle_k) = \langle \mathcal{L}\mathbf{A} \rangle_k \tag{10.30}$$

Now, the mapping of oriented multivector spaces of different maximum grade will be outlined (which is called *projective split* by Hestenes [39]). This is much more than the subspace philosophy of vector algebra. Although it could be successfully applied with respect to *projective geometry* and *kinematics* (see Section 10.3.9), its great potential has not been widely recognized yet. Given a vector space V_n and another V_{n+1} and their respective geometric algebras G_n and G_{n+1}, then any vector $\boldsymbol{X} \in \langle G_{n+1} \rangle_1$ can be related to a corresponding vector $\boldsymbol{x} \in V_n$ with respect to a further unit vector $\boldsymbol{e} \in V_{n+1}, \boldsymbol{e}^2 = 1$, by

$$\mathbf{X}\boldsymbol{e} = \mathbf{X} \cdot \boldsymbol{e}(1 + \boldsymbol{x}) \tag{10.31}$$

or

$$\boldsymbol{x} = \frac{\mathbf{X} \wedge \boldsymbol{e}}{\mathbf{X} \cdot \boldsymbol{e}} \tag{10.32}$$

Thus, the introduction of the reference vector $\boldsymbol{e}$ resembles the representation of $\boldsymbol{x}$ by *homogeneous coordinates* in G_{n+1}, and it includes this important methodology. Yet it goes beyond and can be used to linearize nonlinear transformations and to relate projective, affine, and metric geometry in a consistent manner. Besides, Hestenes [39] introduced the so-called conformal split to relate G_n and G_{n+2} via a unit 2-blade.

10.3.7 Qualitative operations with multivectors

So far we have interpreted the inner and outer products as *quantitative operations*. We also have seen that both the duality operation and the projective split are *qualitative operations*. While the first one realizes a mapping of multivector subspaces with respect to a complementary basis, the second one relates multivector spaces which differ in dimension by one.

With the interesting properties of raising and lowering the grade of multivectors, the outer and inner products also possess qualitative aspects. This is the reason why in geometric algebra higher-order (geometric or kinematic) entities can be simply constructed and their relations can be analyzed, in sharp contrast to vector algebra. Because we so far do not need any metrics of the algebra, the set algebraic aspects of vector spaces should be related to those of the GA. The union ($\cup$) and intersection ($\cap$) of blades, respectively, their vector subspaces (see Eq. (10.11)), goes beyond their meaning in vector algebra. An equation such as

$$\langle G_n(\mathbf{A} \wedge \mathbf{B}) \rangle_1 = \langle G_n(\mathbf{A}) \rangle_1 \cup \langle G_n(\mathbf{B}) \rangle_1 \tag{10.33}$$

where $\mathbf{A}, \mathbf{B}$ are blades, has to be interpreted as an oriented union of oriented vector subspaces [2]. With respect to the basic operations of the *incidence algebra*, that means the meet and the join operations, the extended view of GA will be best demonstrated. The *join* of two blades of grade r and s

$$\mathbf{C} = \mathbf{A} \bigwedge \mathbf{B} \tag{10.34}$$

is their common dividend of lowest grade. If $\mathbf{A}$ and $\mathbf{B}$ are linear independent blades, the join and the outer product are operations of the same effect, thus $\mathbf{C}$ is of grade $r+s$. Otherwise, if $\mathbf{A} \wedge \mathbf{B} = 0$, the join represents the spanned subspace. The *meet*, on the other hand, $\mathbf{C} = \mathbf{A} \bigvee \mathbf{B}$, is indirectly defined with respect to the join of both blades by

$$\mathbf{C}^* = (\mathbf{A} \bigvee \mathbf{B})^* = \mathbf{A}^* \bigwedge \mathbf{B}^* \tag{10.35}$$

It represents the common factor of $\mathbf{A}$ and $\mathbf{B}$ with greatest grade. In case of $r + s = n$ the meet is of grade $|r - s|$ and will be expressed by

$$\mathbf{C} = \boldsymbol{A}^* \cdot \boldsymbol{B} \tag{10.36}$$

Because the Grassmann algebra is based on both the *progressive product* and the *regressive product* of the incidence algebra, it does not surprise that both GA and Grassmann algebra are useful for *projective geometry*, although the projective split and the duality principle of GA enable a more intuitive geometric view and help enormously in algebraic manipulations.

10.3.8 Geometric algebra of the Euclidean space

In this subsection we will consider the GAs of the Euclidean space E_3 and of the Euclidean plane E_2.

From now on we have to consider metric GAs because their vector spaces are metrical ones. A vector space V_n should be endowed with a *signature* (p, q, r) with $p + q + r = n$, so that $G_{p,q,r}$ indicates the number of basis vectors $\boldsymbol{e}_i$, $i = 1, \ldots, n$ that square in the following way

$$\boldsymbol{e}_{i_j}^2 = \begin{cases} 1 & \text{if} \quad j \in \{1, \ldots, p\} \\ -1 & \text{if} \quad j \in \{p+1, \ldots, p+q\} \\ 0 & \text{if} \quad j \in \{p+q+1, \ldots, p+q+r\} \end{cases} \tag{10.37}$$

We call an algebra with $r \neq 0$ a *degenerated* algebra because those basis vectors with $\boldsymbol{e}_{i_j}^2 = 0$ do not contribute to a *scalar product* of the vector space. This results in a constraint of viewing the vector space on a submanifold. In case of $r = 0$ we omit this component and will write $G_{p,q}$. Furthermore, if additionally $q = 0$, the vector space will have a *Euclidean metric*.

For a given dimension n of a vector space V_n there are GAs that are algebraically isomorphic (but not geometrically equivalent), for example, $G_{1,1}$ and $G_{2,0}$, while others are not isomorphic (as $G_{0,2}$ to the other two). In any case, the partitioning $V_n = V_p \cup V_q \cup V_r$ will result in the more or less useful properties of the corresponding algebra $G_{p,q,r}$. We will consider a constructive approach of GAs resulting from the projective split Eq. (10.31) and the separation of an algebraic space into an even and an odd linear subspace following Eq. (10.14). While $G_{p,q}^-$ contains the original vector space by $V_n = \langle G_{p,q} \rangle_1$, there exists an *algebra isomorphism* with respect to $G_{p,q}^+$ [39]

$$G_{p',q'} = G_{p,q}^+ \tag{10.38}$$

The projective split results for a given unit 1-blade $\boldsymbol{e}$ in $p' = q$, $q' = p - 1$ in case of $\boldsymbol{e}^2 \geq 0$, respectively, $p' = p$, $q' = q - 1$ for $\boldsymbol{e}^2 \leq 0$. We will discuss some basic results for $E_3 = \mathbb{R}^3$ and will consider $G_{3,0}$:

Example 10.1: ***Geometric Algebra of the Euclidean Space***

dim $(G_{3,0}) = 8$ (1 scalar, 3 vectors, 3 bivectors, 1 pseudoscalar)
basis $(G_{3,0})$: $\{1, \boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_{23}, \boldsymbol{e}_{31}, \boldsymbol{e}_{12}, \boldsymbol{e}_{123} \equiv I\}$
$\boldsymbol{e}_1^2 = \boldsymbol{e}_2^2 = \boldsymbol{e}_3^2 = 1$, $\boldsymbol{e}_{23}^2 = \boldsymbol{e}_{31}^2 = \boldsymbol{e}_{12}^2 = -1$, $\boldsymbol{e}_{123}^2 = I^2 = -1$
$G_{3,0}^+ \simeq \mathbb{H}$
dim $(G_{3,0}^+) = 4$
basis $(G_{3,0}^+)$: $\{1\ , \boldsymbol{e}_{23}, \boldsymbol{e}_{31}, \boldsymbol{e}_{12}\}$

Here $\boldsymbol{e}_{ij} = \boldsymbol{e}_i\boldsymbol{e}_j = \boldsymbol{e}_i \wedge \boldsymbol{e}_j$ and $\boldsymbol{e}_{123} = \boldsymbol{e}_1\boldsymbol{e}_2\boldsymbol{e}_3$. Application of the duality principle Eq. (10.18) results in $\boldsymbol{e}_{23} = I\boldsymbol{e}_1 \equiv$ -$\boldsymbol{i}$, $\boldsymbol{e}_{31} = I\boldsymbol{e}_2 \equiv \boldsymbol{j}$, and $\boldsymbol{e}_{12} = I\boldsymbol{e}_3 \equiv \boldsymbol{k}$. This 4-D linear space is algebraically isomorphic to the *quaternion algebra*, thus $G_{3,0}^+ \equiv \mathbb{H}$.

While quaternions are restricted to $\mathbb{R}^3$, the same principle applied to $\mathbb{R}^2$ will result in an algebraic isomorphism to *complex numbers* $\mathbb{C}$.

Example 10.2: ***Geometric Algebra of the Euclidean Plane***

dim $(G_{2,0}) = 4$ (1 scalar, 2 vectors, 1 pseudoscalar)
basis $(G_{2,0})$: $\{1, \boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_{12} \equiv I\}$
$\boldsymbol{e}_1^2 = \boldsymbol{e}_2^2 = 1$, $\boldsymbol{e}_{12}^2 = I^2 = -1$
$G_{2,0}^+ \simeq \mathbb{C}$
dim $(G_{2,0}^+) = 2$
basis $(G_{2,0}^+)$: $\{1, \boldsymbol{e}_{12}\}$

Here the imaginary unit $\boldsymbol{i}$ is either a bivector $\boldsymbol{e}_{12}$ or a pseudoscalar I. While in the first case it represents the unit of the directed area embedded in $\mathbb{R}^2$, it indicates in the second case that the unit pseudoscalar is orthogonal to the unit scalar. Thus, any $\boldsymbol{x} \in \mathbb{R}^2$, $\boldsymbol{x} = x_1\boldsymbol{e}_1 + x_2\boldsymbol{e}_2$, may also be represented as complex number $\mathbf{S} \in \mathbb{C}, \mathbf{S} = x_1 + x_2\boldsymbol{i}$. If $\boldsymbol{i}$ is interpreted as a bivector, the term $\boldsymbol{S}$ will also be called *spinor*.

Any bivector $\mathbf{B} \in G_{3,0}$, $\mathbf{B} = \boldsymbol{a} \wedge \boldsymbol{c}$, may either be represented with respect to the bivector basis

$$\mathbf{B} = B_{23}\boldsymbol{e}_{23} + \mathbf{B}_{31}\boldsymbol{e}_{31} + \mathbf{B}_{12}\boldsymbol{e}_{12} \tag{10.39}$$

as the *pure quaternion* or *vector quaternion*

$$\mathbf{B} = B_{23}\boldsymbol{i} + B_{31}\boldsymbol{j} + B_{12}\boldsymbol{k} \tag{10.40}$$

or as the *dual vector*

$$\mathbf{B} = I\boldsymbol{b} = I(b_1\boldsymbol{e}_1 + b_2\boldsymbol{e}_2 + b_3\boldsymbol{e}_3) \tag{10.41}$$

with $b_1 = B_{23}$, $b_2 = B_{31}$, $b_3 = B_{12}$. The bivector coordinates

$$B_{ij} = (\boldsymbol{a} \wedge \boldsymbol{c})_{ij} = a_i c_j - a_j c_i \tag{10.42}$$

represent an antisymmetric tensor of rank two. Because $\boldsymbol{b} = \mathbf{B}^*$ is orthogonal to $\mathbf{B}$, it corresponds to the vector *cross product* in $G_{3,0}$

$$\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{c} = -I(\boldsymbol{a} \wedge \boldsymbol{c}) \tag{10.43}$$

which is defined in $\mathbb{R}^3$.

10.3.9 Projective and kinematic spaces in geometric algebra

Here we will consider non-Euclidean deformations of the space E_3, which are useful to transform nonlinear problems to linear ones. This can be done by an *algebraic embedding* of E_3 in an extended vector space $\mathbb{R}^4$. In case of the *projective geometry* the result will be the *linearization* of *projective transformations*, and in case of the *kinematics* the *linearization* of the *translation* operation.

If we consider a 3-D *projective space* P_3, it may be extended to a 4-D space $\mathbb{R}^4$ using the *projective split* with respect to a unit 1-blade $\boldsymbol{e}_4$, $\boldsymbol{e}_4^2 = 1$. Thus, $G_{1,3}$ is the GA of the 3-D projective space. Its properties are

Example 10.3: ***Geometric Algebra of the Projective Space***

dim $(G_{1,3}) = 16$ (1 scalar, 4 vectors, 6 bivectors, 4 trivectors, 1 pseudoscalar)
basis $G_{1,3}$: $\{1, \boldsymbol{e}_k, \boldsymbol{e}_{23}, \boldsymbol{e}_{31}, \boldsymbol{e}_{12}, \boldsymbol{e}_{41}, \boldsymbol{e}_{42}, \boldsymbol{e}_{43}, I\boldsymbol{e}_k, \boldsymbol{e}_{1234} = I;$ $k = 1, 2, 3, 4\}$
$\boldsymbol{e}_j^2 = -1$, $j = 1, 2, 3$, $\boldsymbol{e}_4^2 = 1$, $\boldsymbol{e}_{1234}^2 = I^2 = -1$

This formulation leads to a *Minkowski metric* of the space $\mathbb{R}^4$ [40]. The projective split with respect to the homogeneous coordinate vector $\boldsymbol{e}_4$ relates the entities of $\mathbb{R}^4$ to their representations in E_3 such that any $\boldsymbol{x} \in E_3$ can be represented by any $\mathbf{X} \in \mathbb{R}^4$ by

$$\boldsymbol{x} = \frac{\mathbf{X} \wedge \boldsymbol{e}_4}{X_4} \tag{10.44}$$

The coordinate $\boldsymbol{e}_4$ corresponds to the direction of the projection [38].

Finally, we will consider the extension of the Euclidean space E_3 to a 4-D *kinematic space* $\mathbb{R}^4$ using the projective split with even the same unit 1-blade $\boldsymbol{e}_4$, but $\boldsymbol{e}_4^2 = 0$. Thus, $G_{3,0,1}$ will have the following properties:

Example 10.4: ***Geometric Algebra of the Kinematic Space***

dim $(G_{3,0,1}) = 16$ (1 scalar, 4 vectors, 6 bivectors, 4 trivectors, 1 pseudoscalar)
basis $(G_{3,0,1}) : \{1, \boldsymbol{e}_k, \boldsymbol{e}_{23}, \boldsymbol{e}_{31}, \boldsymbol{e}_{12}, \boldsymbol{e}_{41}, \boldsymbol{e}_{42}, \boldsymbol{e}_{43}, I\boldsymbol{e}_k, \boldsymbol{e}_{1234} = I;$
$k = 1, 2, 3, 4\}$
$\boldsymbol{e}_j^2 = 1,\ j = 1, 2, 3\ ,\ \boldsymbol{e}_4^2 = 0,\ \boldsymbol{e}_{1234}^2 = I^2 = 0$
$G_{3,0,1}^+ \simeq \mathbb{H} + I\mathbb{H}$
dim $(G_{3,0,1}^+) = 8$
basis $(G_{3,0,1}^+) : \{1, \boldsymbol{e}_{23}, \boldsymbol{e}_{31}, \boldsymbol{e}_{12}, \boldsymbol{e}_{41}, \boldsymbol{e}_{42}, \boldsymbol{e}_{43}, \boldsymbol{e}_{1234} = I\}$

It can be recognized that the bivector basis of $G_{3,0,1}^+$ has to be divided into two groups $\{\mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12}\}$ and $\{\mathbf{e}_{41}, \mathbf{e}_{42}, \mathbf{e}_{43}\}$ with respect to a duality operation. From this it results that the basis can be built by two sets of quaternions, the one dual to the other. The synthesized algebra is isomorphic to the *algebra of dual quaternions* $\hat{\mathbb{H}}$ [31, 41]

$$G_{3,0,1}^+ \simeq \mathbb{H} + I\mathbb{H} \equiv \hat{\mathbb{H}} \tag{10.45}$$

with $I^2 = 0$.

This algebra is of fundamental importance for handling *rigid displacements* in 3-D space as *linear transformations* [32, 33, 34]. This intuitively follows from the property of a unit quaternion coding general 3-D rotation of a point, represented by a vector, around another point as center of rotation. Instead, in $G_{3,0,1}^+$ there are two lines, related by a rigid displacement in space. Thus, the basic assumption in the presented extension of $G_{3,0}^+$ to $G_{3,0,1}^+$ is to use the representation of the 3-D space by lines instead of points [38, 42]. The resulting geometry is therefore called *line geometry*, respectively, *screw geometry*. The last term results from the fact that the axis of a screw displacement will be an invariant of rigid displacements in $G_{3,0,1}^+$, just as the point that represents the center of a general rotation will be the invariant of this operation in $G_{3,0}^+$.

10.3.10 Geometric entities in geometric algebra

We will express basicgeometric entities in the algebraic languages of the Euclidean, projective and kinematic spaces, respectively. Geometric *entities* are *points*, *lines*, and *planes* as the basic units or *irreducible invariants* of yet higher-order agglomerates. The goal will not be to construct a world of polyhedrons. But the mentioned entities are useful for modeling local processes of perception and action, Section 10.3.1, because of the low dimensionality of the local state space.

In the following we will denote points, lines, and planes by the symbols **X**, **L** , and **E**, respectively, if we mean the entities, which we want to express in any GA.

First, to open the door of an *analytic geometry* (see, e.g., Hestenes [29, chapter 2-6.]), we will look at the entities of $G_{3,0}$. In this algebra the identity

$$\mathbf{X} = \boldsymbol{x} \tag{10.46}$$

$\mathbf{X} \in \langle G_{3,0} \rangle_1, \boldsymbol{x} \in E_3$ is valid. The *point* conception is the basic one and all other entities are aggregates of points. Thus,

$$\boldsymbol{x} \wedge I = 0 \tag{10.47}$$

implicitly defines E_3 by the unit pseudoscalar expressing its collinearity with all $\boldsymbol{x} \in E_3$.

The same principle can be used to define a *line* $\boldsymbol{l}$ through the origin by the nonparametric equation $\boldsymbol{x} \wedge \boldsymbol{l} = 0$, therefore, $\boldsymbol{l} = \lambda \bar{\boldsymbol{l}}$ is the set $\{\boldsymbol{x}\}$ of points belonging to the line, and

$$\mathbf{L} = \boldsymbol{l} \tag{10.48}$$

tells us that all such lines are 1-blades $\mathbf{L} \in \langle G_{3,0} \rangle_1$ of direction $\bar{\boldsymbol{l}}$ and $\lambda \in \langle G_{3,0} \rangle_0$ now defines all points belonging to that subspace.

A more general definition of directed lines, not passing the origin, is based on Hesse's normal form. In this case the line is an inhomogeneous multivector, consisting of the vector $\boldsymbol{l}$ and the bivector **M**

$$\mathbf{L} = \boldsymbol{l} + \mathbf{M} = (1 + \boldsymbol{d})\boldsymbol{l} \tag{10.49}$$

Because Eq. (10.49) is in GA, we have a geometric product on the right side. While the vector $\boldsymbol{l}$ is specifying the *direction* of the line, the bivector $\mathbf{M} = \boldsymbol{d}\boldsymbol{l} = \boldsymbol{d} \wedge \boldsymbol{l}$ specifies its *moment*. From the definition of the *moment* we see that $\boldsymbol{d}$ is a vector orthogonal to $\boldsymbol{l}$, directed from origin to the line. It is the minimal *directed distance* of the line from the origin. In the hyperplane spanned by **M** will also lie the normal vector of the line $\boldsymbol{l}^{-1}$, so that $\boldsymbol{d} = \mathbf{M}\boldsymbol{l}^{-1} = \mathbf{M} \cdot \boldsymbol{l}^{-1}$. Besides, we recognize that both $\boldsymbol{l}$ and $\mathbf{M}$ or $\boldsymbol{l}$ and $\boldsymbol{d}$ are *invariants* of the subspace line. From this it follows that all points $\boldsymbol{x} \in \mathbf{L}$ will parametrically define the line by $\boldsymbol{x} = (\mathbf{M} + \lambda)\boldsymbol{l}^{-1}$.

With the same model of Hesse's normal form we can express the plane $\mathbf{E}$ by the 2-blade *direction* $\mathbf{P}$ and the *moment* $\mathbf{M} = \boldsymbol{d}\mathbf{P} = \boldsymbol{d} \wedge \mathbf{P}$, $\mathbf{M} \in \langle G_{3,0} \rangle_3$,

$$\mathbf{E} = \mathbf{P} + \mathbf{M} = (1 + \boldsymbol{d})\mathbf{P} \tag{10.50}$$

Here again $\boldsymbol{d} = \mathbf{M}\mathbf{P}^{-1} = \mathbf{M} \cdot \mathbf{P}^{-1}$ is the *directed distance* of the plane to the origin and all points belonging to the subspace plane are constrained by $\boldsymbol{x} = (\mathbf{M} + \lambda)\mathbf{P}^{-1}$.

Second, we will demonstrate the *incidence algebra* of points, lines and planes in *projective geometry* $G_{1,3}$.

In $G_{3,0}$ equations such as $\boldsymbol{x} \wedge \boldsymbol{l} = 0$ or $\boldsymbol{x} \wedge (\boldsymbol{l} + \mathbf{M}) = 0$ specify the incidence of any point $\boldsymbol{x}$ and line $\boldsymbol{l}$, respectively, line $\mathbf{L}$. In $G_{1,3}$ the operations joint and meet will result in the corresponding incidences. Let $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3 \in \langle G_{1,3} \rangle_1$ be noncollinear points. Any $\mathbf{X} \in \langle G_{1,3} \rangle_1$ will be related with $\boldsymbol{x} \in \mathbb{R}^3$ by the *projective split*. Following the *join* Eq. (10.34), we construct a *line* $\mathbf{L} \in \langle G_{1,3} \rangle_2$ by

$$\mathbf{L}_{12} = \mathbf{X}_1 \bigwedge \mathbf{X}_2 \tag{10.51}$$

and a *plane* $\mathbf{E} \in \langle G_{1,3} \rangle_3$ by

$$\mathbf{E}_{123} = \mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \mathbf{X}_3 \tag{10.52}$$

Each point $\mathbf{X}$ on a line $\mathbf{L}$ results in $\mathbf{X} \wedge \mathbf{L} = 0$, and each point $\mathbf{X}$ on a plane $\mathbf{E}$ results in $\mathbf{X} \wedge \mathbf{E} = 0$, thus $\mathbf{L}$ and $\mathbf{E}$ specify subspaces. The intersections of entities result from the *meet* operation Eq. (10.36). If $\mathbf{X}_s$ is a *point of intersection* of a plane $\mathbf{E}$ and a noncollinear line $\mathbf{L}$ we get, for example,

$$\mathbf{X}_s = \mathbf{L} \bigvee \mathbf{E} = \sigma_1 \mathbf{Y}_1 + \sigma_2 \mathbf{Y}_2 + \sigma_3 \mathbf{Y}_3 \tag{10.53}$$

with $\mathbf{E} = \mathbf{Y}_1 \bigwedge \mathbf{Y}_2 \bigwedge \mathbf{Y}_3$, $\mathbf{L} = \mathbf{X}_1 \bigwedge \mathbf{X}_2$, and $\sigma_1, \sigma_2, \sigma_3 \in \langle G_{1,3} \rangle_0$ are the *brackets* of pseudoscalars of each four points:

$$\sigma_1 = [\mathbf{X}_1\mathbf{X}_2\mathbf{Y}_2\mathbf{Y}_3], \quad \sigma_2 = [\mathbf{X}_1\mathbf{X}_2\mathbf{Y}_3\mathbf{Y}_1], \quad \text{and} \quad \sigma_3 = [\mathbf{X}_1\mathbf{X}_2\mathbf{Y}_1\mathbf{Y}_2]$$

The intersection of two noncollinear planes $\mathbf{E}_1 = \mathbf{X}_1 \bigwedge \mathbf{X}_2 \bigwedge \mathbf{X}_3$ and $\mathbf{E}_2 = \mathbf{Y}_1 \bigwedge \mathbf{Y}_2 \bigwedge \mathbf{Y}_3$ will result in the *intersection line*

$$\mathbf{L}_s = \mathbf{E}_1 \bigvee \mathbf{E}_2 = \sigma_1(\mathbf{Y}_2 \bigwedge \mathbf{Y}_3) + \sigma_2(\mathbf{Y}_3 \bigwedge \mathbf{Y}_1) + \sigma_3(\mathbf{Y}_1 \bigwedge \mathbf{Y}_2) \tag{10.54}$$

with $\sigma_1 = [\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{Y}_1]$, $\sigma_2 = [\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{Y}_2]$ and $\sigma_3 = [\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{Y}_3]$. The derivations have been omitted, see, for example, [43]. On the base of this incidence algebra the constraints resulting from two or three cameras on a geometrically intuitive way can be developed [43].

Finally, we will summarize the representation of points, lines, and planes as entities of the *algebra of kinematics* $G^+_{3,0,1}$, with respect to their constituents in $\mathbb{R}^3$. We learned that in this case the Euclidean space was modeled by the set of all possible lines. From this results the importance of the bivector base of $G^+_{3,0,1}$. Besides, we identified the dualities $\boldsymbol{e}_{41} = I\boldsymbol{e}_{23}, \boldsymbol{e}_{42} = I\boldsymbol{e}_{31}, \boldsymbol{e}_{43} = I\boldsymbol{e}_{12}$, which resulted in a *dual quaternion algebra*. Thus, also the desired entities should have dual quaternion structure $\hat{\mathbf{Z}} = \mathbf{Z} + I\mathbf{Z}^*$ with $\mathbf{Z}, \mathbf{Z}^* \in \mathbb{H}$ and $\hat{\mathbf{Z}} \in \hat{\mathbb{H}}$, $\mathbf{Z} = (Z_1, Z_2, Z_3, Z_4)$, $\mathbf{Z}^* = (Z_1^*, Z_2^*, Z_3^*, Z_4^*)$, so that $\hat{Z}_i = Z_i + IZ_i^*$ are *dual numbers*, from which $\hat{\mathbf{Z}}$ may be represented as 4-tuple of dual numbers $\hat{\mathbf{Z}} = (\hat{Z}_1, \hat{Z}_2, \hat{Z}_3, \hat{Z}_4)$. The components $i = 1, 2, 3$ are from $\mathbb{R}^3$ and, therefore, are representing the vector part of a quaternion, $i = 4$ should indicate the scalar component of a quaternion. From this it follows that any $\hat{\mathbf{X}} \in G^+_{3,0,1}$ should be constraint to the hyperplane $X_4 = 1$, even as the other entities $\hat{\mathbf{L}}$ and $\hat{\mathbf{E}}$.

A *line* $\hat{\mathbf{L}} = \hat{\mathbf{X}}_1 \wedge \hat{\mathbf{X}}_2$, $X_{14} = X_{24} = 1$, may be expressed by

$$\hat{\mathbf{L}} = \mathbf{L} + I\mathbf{L}^* \tag{10.55}$$

$I^2 = 0$, where the scalar components of $\mathbf{L}$ and $\mathbf{L}^*$ are *Plücker coordinates* of the line $\hat{\mathbf{L}}$ that also define the coordinate bivector of a plane through the origin of $\mathbb{R}^4$. This plane is spanned by the moment $\mathbf{M} = \boldsymbol{d}\mathbf{L}$ that intersects E_3, defined as a hyperplane in $\mathbb{R}^4$, in the line $\hat{\mathbf{L}}$. Now, we can identify $\mathbf{L}$ as the *line direction*, represented as an element of $\mathbb{H}$ by

$$\mathbf{L} = L_{23}\boldsymbol{e}_{23} + L_{31}\boldsymbol{e}_{31} + L_{12}\boldsymbol{e}_{12} \tag{10.56}$$

and its dual part by the *moment*

$$\mathbf{L}^* = \mathbf{M} = L_{41}\boldsymbol{e}_{23} + L_{42}\boldsymbol{e}_{31} + L_{43}\boldsymbol{e}_{12} \tag{10.57}$$

thus

$$\hat{\mathbf{L}} = \mathbf{L} + I\mathbf{M} \tag{10.58}$$

For the representation of a *point* $\mathbf{X}$ in line geometry we choose two points $\hat{\mathbf{X}}_1 = (0, 0, 0, 1)$ and $\hat{\mathbf{X}}_2 = (X_1, X_2, X_3, 1)$. This results in

$$\hat{\mathbf{X}} = 1 + I\mathbf{X} \tag{10.59}$$

where $\mathbf{X}$ is identified as the point $\hat{\mathbf{X}}_2$ on the hyperplane $X_4 = 1$.

Finally, let us consider a *plane* $\hat{\mathbf{E}}$ as a tangential plane at a point $\hat{\mathbf{X}}$ that is collinear with the line and orthogonal to its *moment*. We get

$$\hat{\mathbf{E}} = \mathbf{P} + I\mathbf{M} \tag{10.60}$$

where $\mathbf{P}$ is the 2-blade *direction* of Eq. (10.50) and $I\mathbf{M} = \mathbf{D}$ determines the 2-blade directed distance to the origin.
At this point we summarize that the considered entities assume different representations in dependence of the interesting aspects. By switching the signature of the embedding algebra, the one or another aspect comes into play. But if we consider the objects of interest in computer vision as moving rigid bodies, projective and kinematics interpretations have to be fused in a common algebraic approach. This will be a matter of future research.

10.3.11 Operational entities in geometric algebra

Operational entities are geometric transformations that code the net movements of any geometric entity. The simplest ones are *reflection*, *rotation*, *translation* and *scaling* of points. A general principle to get the complete set of primitive geometric operations in E_3 is given by the *Lie group operations*. This space of geometric transformations is of dimension 6. In the given context, first of all we are only interested in a subset related to rotation and translation. If rigid point agglomerations are of interest, or if the degrees of freedom increase because of increasing dimension of the space $\mathbb{R}^n$, also these primitive operations will become complicated, time consuming, or they will assume nonlinear behavior. Because these operations mostly are of interest in combinations, the question is if there are algebraic embeddings in which these combine to new ones with own right to be considered as basic operations. An example is *rigid displacement*. There has been considerable work in robotics to reach low symbolic complexity for coding the movement of complex configurations and simultaneously to reach low numeric complexity [33, 44, 45, 46]. In the frame of PAC, in addition the fusion of geometric models for geometric signal deformations and robot actions becomes an important problem, for example, with respect to *camera calibration* [47], *pose estimation*, or *gesture tracking* [48]. We call the geometric transformations entities because they themselves are represented by multivectors in GA. Of course, their special form and their properties depend strongly on the special algebraic embedding of the problem.

First, we will consider $G_{3,0}$, see Example 10.1, and $G_{2,0}$, see Example 10.2. Because of Eqs. (10.3) and (10.27), any two vectors $\boldsymbol{a}, \boldsymbol{b} \in G_{2,0}$ result in an inhomogeneous multivector

$$\mathbf{C} = \boldsymbol{ab} = \cos\theta + I\sin\theta = \exp(I\theta) \tag{10.61}$$

where θ is the radian measure of the enclosed angle and the bivector $\Theta = I\theta$ describes an angular relation between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ in the plane. Indeed, Θ is a directed area, circularly enclosed between $\boldsymbol{a}$ and

$\boldsymbol{b}$, whose direction indicates a rotation from $\boldsymbol{a}$ to $\boldsymbol{b}$. A rotation from $\boldsymbol{b}$ to $\boldsymbol{a}$ would result in the reversed multivector

$$\tilde{\mathbf{C}} = \boldsymbol{ba} = \cos\theta - I\sin\theta = \exp(-I\theta) \tag{10.62}$$

Any rotated vectors $\boldsymbol{a}$ and $\boldsymbol{a}'$ are related by either $\boldsymbol{a}' = \boldsymbol{a}\exp(I\theta)$ in case of an orthogonal rotation or by $\boldsymbol{a}' = \exp(\frac{I\theta}{2})\boldsymbol{a}\exp(-\frac{I\theta}{2})$ in case of a general rotation. We write instead (see Hestenes [29])

$$\boldsymbol{a}' = \mathbf{Ra}\tilde{\mathbf{R}} \tag{10.63}$$

and call

$$\mathbf{R} = \exp\left(\frac{I\theta}{2}\right) \tag{10.64}$$

a *rotor* with $\mathbf{R}\tilde{\mathbf{R}} = 1$.

In comparison to other operational codes of rotation the rotor has several advantages. Its structure as a bivector is the same as that of the objects. It is a linear operator and as such it is the same, whatever the grade of the geometric object and whatever the dimension of the space is. In contrast to matrix algebra a rotor is completely coordinate independent. The linearity results in $\mathbf{R}_3 = \mathbf{R}_2\mathbf{R}_1$, thus

$$\boldsymbol{a}'' = \mathbf{R}_3\boldsymbol{a}\tilde{\mathbf{R}}_3 = \mathbf{R}_2\boldsymbol{a}'\tilde{\mathbf{R}}_2 = \mathbf{R}_2\mathbf{R}_1\boldsymbol{a}\tilde{\mathbf{R}}_1\tilde{\mathbf{R}}_2 \tag{10.65}$$

Because a rotor is a bivector, it will be isomorphic to a unit quaternion in $G^+_{3,0}$. If we couple rotation with scaling, then

$$\mathbf{S} = s + \mathbf{R}, \tag{10.66}$$

$s \in \langle G_n\rangle_0$, will be a *spinor*. Equation (10.66) can be expressed in multiplicative form because of Eq. (10.64). Then the action of a spinor corresponds to the action of another rotor R' that is not of unit magnitude. The algebra $G^+_{3,0}$ is also called *spinor algebra*.

A translation in $\mathbb{R}^n$ corresponds to an n-dimensional vector, say $\boldsymbol{t}$. With respect to a single point $\mathbf{x} \in \mathbb{R}^n$ the translation will be a linear operation because $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}$. But this is not valid if two points $\boldsymbol{x}$ and $\boldsymbol{y}$, related by a distance $\boldsymbol{d} = \boldsymbol{y} - \boldsymbol{x}$, will simultaneously be translated by the same vector because

$$\boldsymbol{d} + \boldsymbol{t} \neq (\boldsymbol{y} + \boldsymbol{t}) - (\boldsymbol{x} + \boldsymbol{t}) \tag{10.67}$$

A rigid transformation preserves the distance between points of any aggregation of such. It results in a rigid displacement of the aggregation. *Rigid transformations* are rotation and translation, both coupled together. With the result of Eq. (10.67) no linear method exists in Euclidean space to perform general rigid displacements by linear transformations.

The introduction of *homogeneous coordinates* results in several different linearizations of rigid transformations. But the best algebraic embedding of the problem is given by the GA of the *kinematic space*, Example 10.4, or *motor algebra* $G^+_{3,0,1}$. In this algebra the rigid transformations are expressed by linear relation of two lines. One *oriented line* is representing a rotation $\mathbf{R}$ using a bivector, another oriented line is representing a translation $\mathbf{T}$, also using a bivector.. This is the model of a *motor* $\hat{\mathbf{C}}$ as introduced by [49] and developed by Bayro-Corrochano et al. [50].

$$\hat{\mathbf{C}} = \hat{\mathbf{X}}_1\hat{\mathbf{X}}_2 + \hat{\mathbf{X}}_3\hat{\mathbf{X}}_4 = \mathbf{R} + I\mathbf{R}' \tag{10.68}$$

where $\mathbf{R}$ is the rotor as introduced in Eq. (10.64), and $\mathbf{R}'$ is another rotor, modified by a *translator*,

$$\mathbf{T} = \exp\left(\frac{\boldsymbol{t}I}{2}\right) = 1 + I\frac{\boldsymbol{t}}{2} \tag{10.69}$$

$\boldsymbol{t} = t_1\boldsymbol{e}_{23} + t_2\boldsymbol{e}_{31} + t_3\boldsymbol{e}_{12}$, see Bayro-Corrochano et al. [50]. The translator can be seen in these terms as a plane of rotation, translated by $\boldsymbol{t}$ from the origin in direction of the rotation axis. The motor degenerates in case of coplanarity of both axes to a rotor, representing pure rotation, else it represents a *screw motion* by

$$\hat{\mathbf{C}} = \mathbf{R} + I\frac{\mathbf{t}}{2}\mathbf{R} = \left(1 + I\frac{\mathbf{t}}{2}\right)\mathbf{R} = \mathbf{TR} \tag{10.70}$$

Notice that Eq. (10.70) represents a motor as an element of the motor algebra and that Eq. (10.68) represents a motor as an element of the dual quaternion algebra, $\hat{\mathbf{C}} \in \hat{\mathbb{H}}$. Although both are isomorphic algebras, the motor algebra offers more explicitly the relation between translation and rotation and, therefore, will be more usefully applied in modeling chained movements. If we apply this code of a rigid transformation to a line $\hat{\mathbf{L}} = \mathbf{L} + I\mathbf{M}$, see Eq. (10.58), then the transformed line will be

$$\hat{\mathbf{L}}' = \mathbf{L}' + I\mathbf{M}' = \hat{\mathbf{C}}\hat{\mathbf{L}}\hat{\tilde{\mathbf{C}}} \tag{10.71}$$

or in full [50]

$$\hat{\mathbf{L}}' = \mathbf{RL\tilde{R}} + I(\mathbf{RL\tilde{R}}' + \mathbf{R}'\mathbf{L\tilde{R}} + \mathbf{RM\tilde{R}}) \tag{10.72}$$

The motor algebra and the isomorphic dual quaternion algebra not only have importance for dense coding of robotic tasks, for example, chained links, but they will also have high impact on the analysis of manifolds such as images and image sequences. In the motor algebra not only linearization of certain transformations will be gained. It also enables us to use a high degree of invariance with respect to the related geometric entities.

10.4 Applications of the algebraic framework

We have motivated the need of a powerful framework of algebraic nature from the viewpoint of designing behavior-based systems and from well-known intrinsic problems, such as limitations of the involved disciplines. Furthermore, we could present a useful framework that was formulated more than one hundred years ago but little noticed to date in engineering applications. This framework is geometric algebra or *Clifford algebra*.

We are developing the application of the algebra with respect to robotics, computer vision, multidimensional signal theory, pattern recognition, and neural computing. In the last years we were able to publish several preliminary results. The actual list of publications and reports can be found on `http://www.ks.informatik.uni-kiel.de`. But in fact, both the shaping of the framework and the application of tools are in their infancy. We will not go into details of applications in computer vision and robotics. Instead, we will introduce two aspects that are related to the use of GA with respect to multidimensional structure analysis and recognition. Both topics stress that the GA will have a great impact on describing manifolds. This is also the field where the global frame of events, entities, or patterns meets the local frame. This local frame is strongly related to Lie algebra and Lie groups. Because every matrix *Lie group* can be realized as a *spin group*, and because spin groups consist of even products of unit vectors—generators of a spin group are bivectors, it does not surprise that GA will give the right frame for a Lie group based design of the local frame, see, for example, [2, chapter 8] and [51, 52].

10.4.1 Image analysis and Clifford Fourier transform

The recognition of patterns in images means capturing the patterns as a whole and comparing these entities with equivalence classes at hand. In image processing as in engineering in general, linear methods as linear transformations or linear filters are preferable to non linear ones because they are easier to design and handle. Unfortunately, linear methods most often are not useful to describe *multidimensional structures* or to design filters that specifically make explicit such structures. In case of global patterns that consist of multiple entities—but which do not have to be distinguished, this does not matter. The Fourier transform, for example, is a global, linear and complete transformation whose power spectrum in such cases can be used to distinguish between several global patterns. In case of linear filters the forementioned problems, quite the reverse, do limit the set of patterns that can be modeled to be cross-correlated. Filters are not universal tools but highly specified ones. They have not to be complete in general, yet complete with respect to the modeled structures.

There are several algebraic approaches to overcome these limitations, for example, the *tensor filter approach* [26, 27] or the *Volterra filter approach* [53, 54, 55]. Both approaches resemble to a certain extent the way we want to sketch. This does not surprise because GA and the other methods are based on related algebraic roots. Yet there are specific differences that should not be discussed here.

In the Volterra series approach of nonlinearity there is an interesting equivalence of the order of nonlinearity and the dimension of the operator, so that the product of both remains constant. That means, if the order of nonlinearity of image processing in dimension N is given by just this dimension and the dimension of an operator should be the same, then the constant would be N^2. To design instead an equivalent filter of first order would necessitate a filter of dimension N^2. The basic origin of the nonlinearity problem in the case of multidimensional LSI operators is that their eigenvectors should correspond with the dimension of the signals. Because the eigenvectors of LSI operators correspond to the basis functions of the Fourier transform, the key for solving the problem is strongly related to the algebraic embedding of the Fourier transform.

These difficulties are also mentioned in textbooks on signal theory—there is no unique definition of a multidimensional phase in Fourier transform. Precisely the phase is the feature that represents the correlations, respectively symmetries of multidimensional structure. It is not a matter of fate to have no multidimensional phase, but a matter of algebraic adequate embedding of the problem. Zetzsche and Barth [56] argued that for 2-D signals the 1-D basis functions of the Fourier transform should be coupled by a logical AND-operator. Later they presented a linear approach of 2-D filtering using the Volterra series method [57]. To prevent 4-D filters, they developed an interesting scheme to operate on a parametric surface in the 4-D space.

Our proposition is that the linear transformation of an N-D signal has to be embedded in a geometric algebra $G_{N,0}$, which means in a linear space of dimension 2^N (see [58]).

We call the *N-D Fourier transform* $\mathbf{F}^N(\boldsymbol{u})$ in GA the *Clifford Fourier transform* (*CFT*); it reads [58]

$$\mathbf{F}^N(\boldsymbol{u}) = \mathcal{F}_c\{f\} = \int \cdots \int f(\boldsymbol{x})\mathbf{C}_{\boldsymbol{u}}^N(\boldsymbol{x})d^N\boldsymbol{x} \qquad (10.73)$$

$\boldsymbol{x}, \boldsymbol{u} \in \mathbb{R}^N$, with the *CFT kernel*

$$\mathbf{C}_{\boldsymbol{u}}^N(\boldsymbol{x}) = \prod_{k=1}^{N} \exp(-2\pi i_k u_k x_k) \qquad (10.74)$$

The inverse CFT is given by

$$f(\boldsymbol{x}) = \mathcal{F}_c^{-1}\{\mathbf{F}^N\}(\boldsymbol{x}) = \int \cdots \int \mathbf{F}^N(\boldsymbol{u})\tilde{\mathbf{C}}_{\boldsymbol{x}}^N(\boldsymbol{u})d^N\boldsymbol{u} \tag{10.75}$$

where the kernel of the inverse CFT

$$\tilde{\mathbf{C}}_{\mathbf{x}}^N(\boldsymbol{u}) = \prod_{k=0}^{N-1} \exp(2\pi i_{N-k}u_{N-k}x_{N-k}) \tag{10.76}$$

is similar to the reverse of a 2^N-vector in the language of GA and corresponds to the conjugate. The CFT kernel spans the 2^N-D space of the GA in case of Fourier transform of an N-D signal. Each 1-D kernel component

$$\mathbf{C}_{u_k}^1(x_k) = \exp(-2\pi i u_k x_k) \tag{10.77}$$

is a bivector (compare Section 10.3.6), $\mathbf{C}_{u_k}^1(x_k) \in G_{2,0}^+$, see Example 10.2.

In the 2-D case we have a quaternionic Clifford kernel $\mathbf{C}_{\boldsymbol{u}}^2(\boldsymbol{x}) \in G_{3,0}^+$ (see Example 10.1),

$$\mathbf{C}_{\mathbf{u}}^2(\boldsymbol{x}) = \mathbf{C}_{u_1}^1(x_1) \wedge \mathbf{C}_{u_2}^1(x_2) \tag{10.78}$$

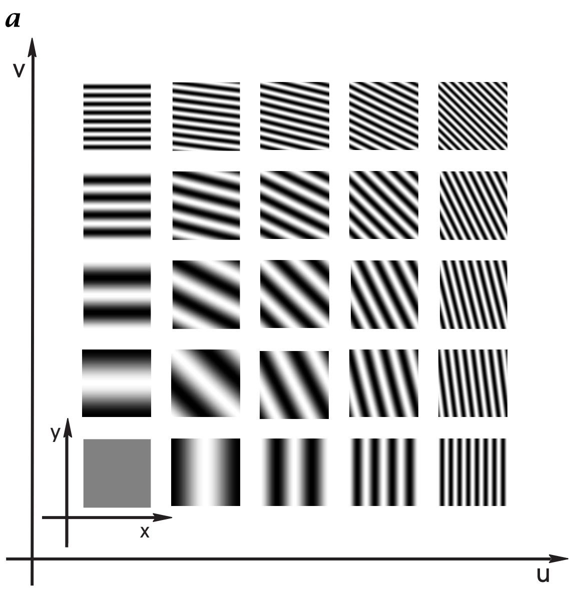
In that algebra exist three different unit 2-blades, which results in enough degrees of freedom for complete representation of all symmetries of the signal.

The 2-D CFT will be called *quaternionic Fourier transform* (QFT). We can generalize the scheme to the preceding proposition about the adequate embedding of N-D Fourier transforms. With this the *Hartley transform* assumes an interesting place in the order scheme [58]. While the Hartley transform covers all symmetries of a signal, the 1-D complex Fourier transform is adequate to represent even and odd *symmetry* in one direction, and the CFT allows for separate representation of even and odd symmetry in orthogonal directions without fusion as in the case of 2-D complex Fourier transform.

This important property becomes visible in Fig. 10.1. There are two nested plots of basis functions of the 2-D Fourier transform in spatial domain with each five frequency samples in orthogonal directions. The basis functions of the complex Fourier transform (top) only represent 1-D structures. Even and odd symmetry occur only in one direction. The basis functions of the quaternionic Fourier transform (bottom) in contrast represent 2-D structures. Even and odd symmetries occur in each 2-D basis function in orthogonal directions, besides in the degenerated cases $\boldsymbol{x} = 0$ or $\boldsymbol{y} = 0$.

Because $\mathbf{F}^2(\boldsymbol{u}) \in G_{3,0}^+$, there are one real and three imaginary components

$$\mathbf{F}^2 = \mathcal{R}(\mathbf{F}^2) + i\mathcal{I}(\mathbf{F}^2) + j\mathcal{J}(\mathbf{F}^2) + k\mathcal{K}(\mathbf{F}^2) \tag{10.79}$$



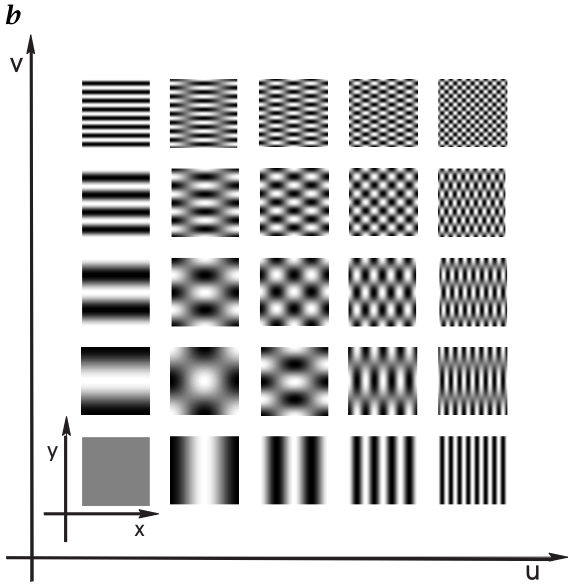


Figure 10.1: *Basis functions of* ***a*** *the 2-D complex Fourier transform; and* ***b*** *the 2-D quaternionic Fourier transform.*

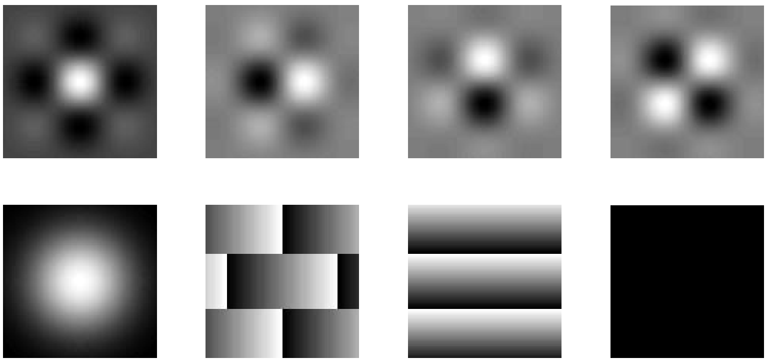


Figure 10.2: *Quaternionic Gabor filter: real and three imaginary components (top); magnitude and three phase angles (bottom).*

which we give a *triple of phase angles* [59]

$$(\Phi, \Theta, \Psi) \in [-\pi, \pi[\times [-\frac{\pi}{2}, \frac{\pi}{2}[\times [-\frac{\pi}{4}, \frac{\pi}{4}] \tag{10.80}$$

This global scheme of algebraic embedding gains its importance from the ability to transfer it to a local scheme, thus to extract the *local phase* of any 2-D structures by means of linear filters. This necessitates formulating the algebraic extension of the analytic signal [60]. The *quaternionic analytic signal* of a real 2-D signal in frequency domain reads

$$\mathbf{F}_A^2(\boldsymbol{u}) = (1 + \operatorname{sign}(u_1))(1 + \operatorname{sign}(u_2))\mathbf{F}^2(\boldsymbol{u}) \tag{10.81}$$

and in spatial domain

$$f_A^2(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{n} \cdot \boldsymbol{f}_{\mathcal{H}i}(\boldsymbol{x}) \tag{10.82}$$

with the Hilbert transform $\boldsymbol{f}_{\mathcal{H}i}(\boldsymbol{x}) = (f_{\mathcal{H}_1}, f_{\mathcal{H}_2}, f_{\mathcal{H}})^T$,

$$f_{\mathcal{H}} = f ** \frac{1}{\pi^2 xy}, \; f_{\mathcal{H}_1} = f ** \frac{\delta(y)}{\pi x}, \; f_{\mathcal{H}_2} = f ** \frac{\delta(x)}{\pi y} \tag{10.83}$$

and $\boldsymbol{n} = (i, j, k)^T$. This scheme of *Hilbert transform* to compute the 2-D analytic signal resembles the results of Hahn [61, 62], but has better algebraic properties, see [60].

Finally, the computation of the local spectral features can be done with an algebraically extended Gabor filter [63]. Figure 10.2 shows on the top row from left to right the real, the i-imaginary, the j-imaginary, and the k-imaginary components of the *quaternionic Gabor filter*. On the bottom row from left to right there is shown the magnitude and

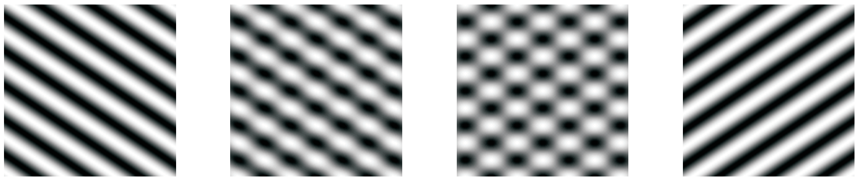


Figure 10.3: *Four texture patterns. Left and right: pure 1-D signals, embedded into 2-D; middle: weighted superpositions to true 2-D signals.*

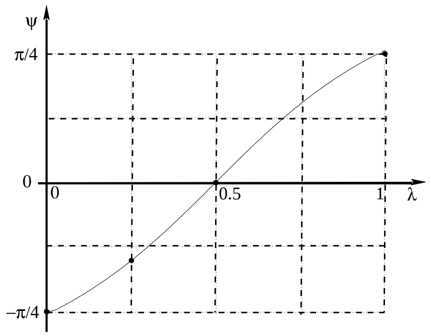


Figure 10.4: *Phase angle Ψ of QFT discriminates the textures of Fig. 10.3.*

the phases Φ, Θ, and Ψ. We will demonstrate the advantage of using the presented scheme of the *2-D phase*. In Fig. 10.3 four textures can be seen that result from superposition of two unbent cosine structures $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ with different spreading directions in 2-D following the rule $\boldsymbol{f}(\lambda) = (1 - \lambda)\boldsymbol{f}_1 + \lambda \boldsymbol{f}_2$. From left to right we have $\lambda = 0, 0.25, 0.5$ and 1. Both complex and quaternionic Gabor filters result in the same horizontal frequency of 0.035 pixel^{-1} and in the same vertical frequency of 0.049 pixel^{-1} for all images. Therefore, no linearly local 1-D analysis could discriminate these patterns. But from the quaternionic phase we get also the parameter Ψ. In Fig. 10.4 we see that the quaternionic phase Ψ well discriminates the textures of Fig. 10.3. It should be noticed that in case of complex Gabor filters four convolutions have been applied in total (two in horizontal and two in vertical direction), while quaternionic Gabor filters need the same number of convolutions. Thus, we conclude that an adequate algebraic embedded design of LSI-operators will result in complete representations of local N-D structures. The development of a linear multidimensional system theory is on the way. A *quaternionic FFT algorithmus* has already been developed.

10.4.2 Pattern recognition and Clifford MLP

Neural networks play a crucial role in designing behavior-based systems. Learning of competence guarantees the designer robustness and invariance of the systems to a certain degree. Within the class of feedforward nets there are two important classes—perceptron derivatives and *radial basis function* (RBF, see Volume 2, Section 23.4) derivatives (for a review of neural networks see Volume 2, Chapter 23). *Multilayer perceptrons* (MLPs, see Volume 2, Section 23.2) have been proven to be the best universal approximators. They operate with a global activation function. Nets of RBF use a local sampling scheme of the manifold. Especially the MLP can be embedded into the geometric algebra [64, 65].

The basic motivation of this research is to design neural nets not only as best universal approximators but as specialists for certain classes of geometric or operational entities. The aim is to find new approaches for the design of PAC systems. The key for reaching this consists of a kind of algebraic blowing up the linear associator because it operates only in the vector space scheme. This means it can handle points only as geometric entities but does not use the whole spectrum of entities with which the linear space of the GA is endowed. The assumption that an MLP can successfully use the additional degrees of freedom of the algebraic coding has been confirmed. Additional degrees of freedom do not only result in better generalization, but accelerate learning. Our limited experience permits interpreting the results as positive ones. We learned that the algebraically extended nets use multivectors both as geometric and operational entities. We call the algebraically embedded MLP a *Clifford MLP* or CMLP . Of course, we know from Section 10.3 that there are multiple algebras. Our design scheme enables us to activate the one or the other, thus we are able to look at the data from several different viewpoints [65].

There is also some research from other groups with respect to design complex and quaternionic neurons, see, for example, [66, 67, 68, 69]. However they developed either special nets for complex or quaternionic numbers or could not handle geometric algebras with *zero divisors* [69]. Our design scheme is bottom up, starting from first principles of geometric algebra and resulting in general CMLPs that use a component wise *activation function* that results in an automorphism that prevents zero divisor problems during backpropagation.

The mapping function of a traditional McCulloch-Pitts neuron for input x, output o, threshold θ and activation function f is given by

$$o = f(\sum_{i=1}^{N} w_i x_i + \theta) \tag{10.84}$$

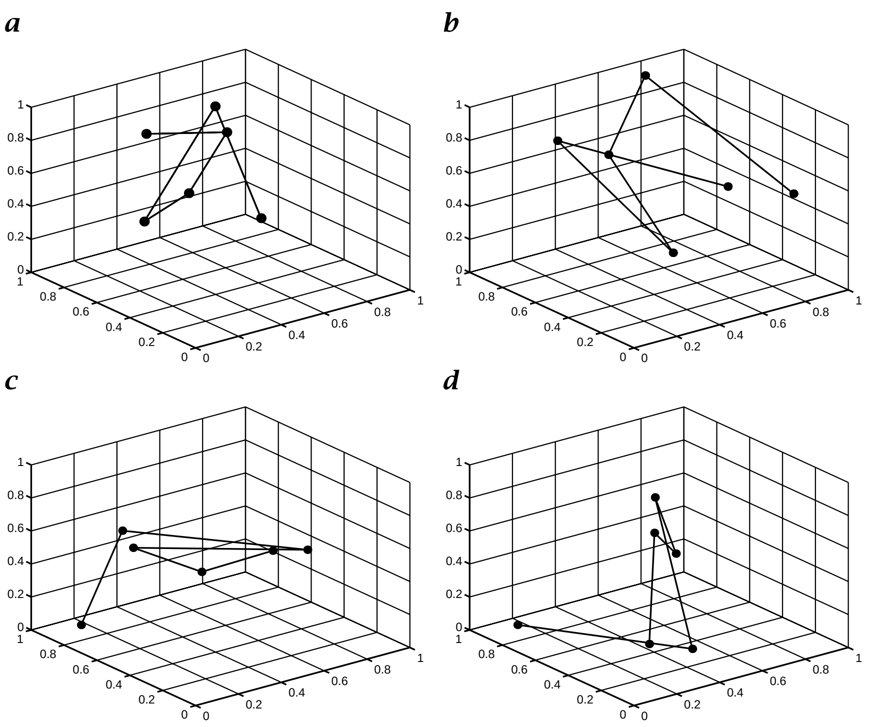


Figure 10.5: Three-dimensional training patterns for MLP- and CMLP-based classification of rigid point agglomerates.

The *Clifford McCulloch-Pitts neuron* [70] on the other hand has the structure

$$\boldsymbol{o} = \boldsymbol{f}(\boldsymbol{w}\boldsymbol{x} + \boldsymbol{\theta}) = \boldsymbol{f}(\boldsymbol{w} \wedge \boldsymbol{x} + \boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{\theta}) \tag{10.85}$$

It contains the scalar product as a vector algebra operation from Eq. (10.84)

$$\boldsymbol{f}(\boldsymbol{w} \cdot \boldsymbol{x} + \theta) = f(\alpha_0) \equiv f(\sum_{i=1}^{N} w_i x_i + \theta) \tag{10.86}$$

and the nonscalar components of any grade of the GA

$$\boldsymbol{f}(\boldsymbol{w} \wedge \boldsymbol{x} + \boldsymbol{\theta} - \theta) = f(\alpha_1)\boldsymbol{e}_1 + f(\alpha_2)\boldsymbol{e}_2 + \cdots + f(\alpha_{2^N-1})\boldsymbol{e}_{1\ldots N} \tag{10.87}$$

To demonstrate the gained performance in comparison to a usual MLP we show in Figure 10.5 four different 3-D figures that are coded with respect to their 3-D point coordinates. The chords connecting two points should demonstrate the shape as 3-D rigid agglomerates of points. The task is simply to learn these patterns with three real hidden neurons of an MLP, respectively, one quaternionic hidden neuron

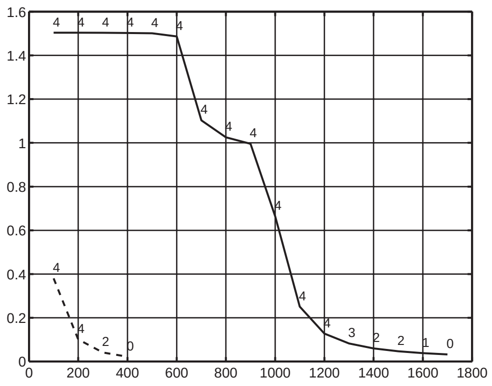


Figure 10.6: *Rate of false classifications by MLP (solid line) and CMLP (broken line) for the patterns of Fig. 10.5.*

of a CMLP, and to gain zero false classifications. In Fig. 10.6 we see the error of learning versus the number of cycles. Besides, the curves (MLP (solid line), CMLP (broken line)) are labeled with the numbers of false classified patterns. Only if the learning error measure drops to approximately 0.02 will both nets gain zero false classifications. Yet the single quaternionic hidden neuron of the CMLP does learn the task with 1/4 the cycles the three real hidden neurons of the MLP need. The performance gain of the CMLP results from its intrinsic capability to represent $G_{3,0}^{+}$. If we would interpret the need of one or three hidden neurons by the two kinds of nets only with respect to representing the components of the 3-D point coordinate vectors, accelerated learning by the CMLP would not result. Instead, the CMLP uses bivectors for representation of the chords between points. These higher-order entities result in an increase of convergence to gain competence as in Fig. 10.6.

Because GA represents not only geometric but also operational entities, it is not surprising that a CMLP is able to learn geometric transformations. We will demonstrate this for a *quaternionic CMLP*, again with one hidden neuron. In Fig. 10.7 we see a patch of connected planes in 3-D space. To the left is the starting pose, to the right the ending pose. The learned transformation is an affine one. The top row is showing the training data with the following parameters of transformation: rotation 45° with respect to axis $[0, 0, 1]$, scaling factor 0.8, translation vector $[0.4, -0.2, 0.2]$. On the bottom row we see the test data, which are changed with respect to the training data by: rotation -60° with respect to axis $[0.5, \sqrt{0.5}, 0.5]$. In Fig. 10.8 we see the comparison of the demanded pose (crosses) with the computed one (solid lines). The quaternionic CMLP uses one hidden neuron to learn exactly any agglomerates of lines with respect to a similarity transformation.

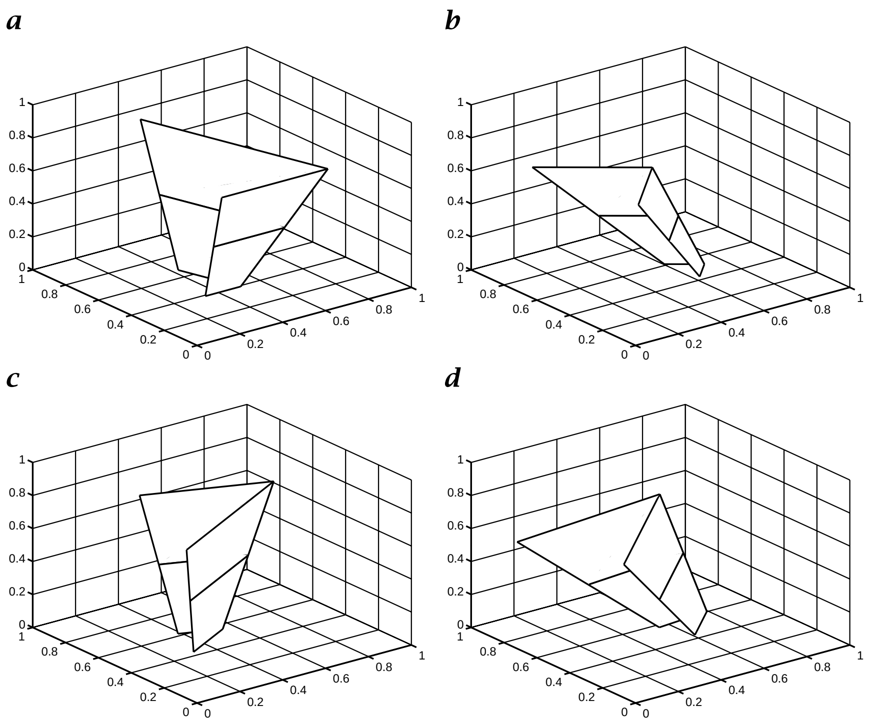


Figure 10.7: *Learning of an affine transformation by a quaternionic CMLP. Top row: training data; bottom row: test data; left column: input data; right column: output data.*

A generalization such as the one shown in Fig. 10.7 cannot be obtained using an MLP with an arbitrary number of neurons. In [68] we can find a complex MLP with an algebraic structure that fits our CMLP on the level of complex numbers. There are also some presentations of learned geometric transformations and the functional analytic interpretation. Our presented frame of algebraic interpretation of the results is more general. It works for all algebraic embeddings. Any Clifford neuron can learn all the group actions that are intrinsic to the corresponding algebra and that are not limited to linear transformations as in the case of a real perceptron.

The presented results demonstrate that algebraic embedded neural nets are worth considering and that we can design on that base a new class of neural nets that constitute an agglomeration of experts for higher-order entities and geometric transformations, respectively. Because both MLPs and CMLPs operate with linear separation functions for each hidden neuron, the algebraic embedding supports this scheme of coding.

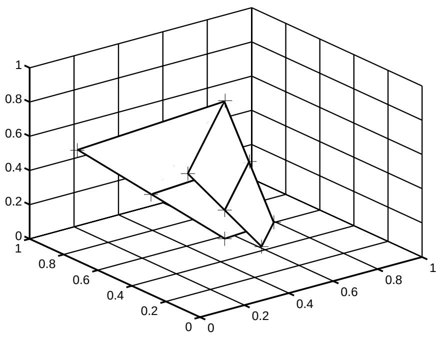


Figure 10.8: *Actual results (solid lines) and demanded results (crosses) of learning an affine transformation by a quaternionic CMLP, see Fig. 10.7.*

10.5 Summary and conclusions

We presented an algebraic language for embedding the design of behavior-based systems. This is geometric algebra, also called Clifford algebra (Clifford originally gave it the name geometric algebra). Our motivation in introducing this language for different disciplines (e.g., computer vision, signal theory, robotics, and neural computing) is to overcome some of the intrinsic problems of the disciplines and to support their fusion in the frame of the perception-action cycle.

We demonstrated the basic ideas of the framework to make obvious both their potential and the demanding task of developing this language to a tool package that is comparable to vector algebra.

Although both the research work and the applications are in their infancy, we presented some examples that can provide us with an impression of the gained attractive performance. We could demonstrate that the embedding of a task into geometric algebra opens the door to linear operations with higher order entities of geometric and kinematic schemes. On that basis, traditional problems may be reformulated with the effects of linearization, higher efficiency, coordinate independency and greater compactness on a symbolic level of coding. As a result of research in this field, one day we can motivate the VLSI design of Clifford processors to gain real profit from the compactness of the language if used for the design of PAC systems.

A dream could become reality: That in the future we have autonomous robots, acting in the complex real world with (visual) percepts and brains that are computers that are capable of representing and manipulating multivectors.

Acknowledgments

The presented research work is supported by DFG grant So 320-2-1 and Studienstiftung des deutschen Volkes.

This chapter was written using research results to which several staff members and many students of the Cognitive Systems Research Group of the Kiel University made contributions. These are E. Bayro-Corrochano, K. Daniilidis (currently in Philadelphia), T. Bülow, S. Buchholz, Yiwen Zang, M. Felsberg, V. Banarer, D. Kähler, and B. Rosenhahn. Our secretary F. Maillard had the hard job of preparing the TEX file.

Our group also thanks J. Lasenby, Cambridge University, for cooperation and D. Hestenes, Tempe, AZ for his stimulating interest in our work.

10.6 References

[1] Clifford, W. K., (1882). *Mathematical Papers.* London: Mcmillan.

[2] Hestenes, D. and Sobczyk, G., (1984). *Clifford Algebra to Geometric Calculus.* Dordrecht: D. Reidel Publ. Comp.

[3] Newell, A. and Simon, H. A., (1976). Computer science as empirical enquiry: symbol and search. *Comm. of the ACM*, **19**:113–126.

[4] Lozano-Perez, T., (1983). Spatial planning: a configuration space approach. *IEEE Trans. Computers*, **32**:108–120.

[5] Marr, D., (1982). *Vision.* San Francisco: W.K. Freeman.

[6] Koenderink, J. J., (1992). Wechsler's vision. *Ecological Psychology*, **4**:121–128.

[7] Hamada, T., (1997). Vision, action and navigation in animals. In *Visual Navigation*, Y. Aloimonos, ed., pp. 6–25. Mahwah, NJ: Lawrence Erlbaum Assoc., Publ.

[8] Fermüller, C. and Aloimonos, Y., (1995). Vision and action. *Image and Vision Computing*, **13**:725–744.

[9] Sommer, G., (1997). Algebraic aspects of designing behavior based systems. In *Algebraic Frames for the Perception-Action Cycle*, G. Sommer and J. Koenderink, eds., Lecture Notes in Computer Science, 1315, pp. 1–28. Berlin, Heidelberg: Springer.

[10] Pauli, J., (1997). Projective invariance and orientation consensus for extracting boundary configurations. In *Mustererkennung 1997*, E. Paulus and F. Wahl, eds., pp. 375–383. Berlin, Heidelberg: Springer.

[11] Michaelis, M. and Sommer, G., (1995). A Lie group approach to steerable filters. *Pattern Recognition Letters*, **16**:1165–1174.

[12] Bruske, J. and Sommer, G., (1997). An algorithm for intrinsic dimensionality estimation. In *Computer Analysis of Images and Patterns*, G. Sommer, K. Daniilidis, and J. Pauli, eds., Lecture Notes in Computer Science, 1296, pp. 9–16. Berlin, Heidelberg: Springer.

[13] Sanger, T. D., (1995). Optimal movement primitives. In *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, eds., Vol. 7, pp. 1023–1030. Cambridge, MA: MIT Press.

[14] Aloimonos, Y. (ed.), (1997). *Visual Navigation*. Mahwah, NJ: Lawrence Erlbaum Assoc., Publ.

[15] Garding, J., Porill, J., Mayhew, J. E. W., and Frisby, J. P., (1995). Stereopsis, vertical disparity and relief transformations. *Vision Research*, **35**:703–722.

[16] Buck, D., (1997). *Projektive Tiefenrepräsentation aus partiell kalibrierten Stereokamerasystemen*. Diploma thesis, Institut für Informatik, Christian-Albrechts-Universität zu Kiel, Kiel, Germany.

[17] Faugeras, O., (1995). Stratification of three-dimensional vision: projective, affine, and metric representations. *Jour. Opt. Soc. Amer.*, **A12**:465–484.

[18] Koenderink, J. J., (1993). Embodiments of geometry. In *Brain Theory*, A. Aertsen, ed., pp. 3–28. Amsterdam: Elsevier Science Publ.

[19] Koenderink, J. J., (1990). The brain a geometry engine. *Psychological Research*, **52**:122–127.

[20] Pellionisz, A. and Llinas, R., (1985). Tensor network theory of the metaorganization of functional geometries in the central nervous system. *Neuroscience*, **16**:245–273.

[21] von der Malsburg, C., (1990). Considerations for a visual architecture. In *Advanced Neural Computers*, R. Eckmiller, ed., pp. 303–312. Amsterdam: Elsevier Science Publ.

[22] Bruske, J. and Sommer, G., (1995). Dynamic cell structure learns perfectly topology preserving map. *Neural Computation*, **7**:845–865.

[23] J. J. Koenderink, A. K. and van Doorn, A., (1992). Local operations: The embodiment of geometry. In *Artificial and Biological Vision Systems*, G. Orban and H.-H. Nagel, eds., ESPRIT Basic Research Series, pp. 1–23. Berlin: Springer.

[24] Danielson, D. A., (1992). *Vectors and Tensors in Engineering and Physics*. Redwood City: Addison-Wesley Publ. Comp.

[25] Frankel, T., (1997). *The Geometry of Physics*. Cambridge: Cambridge University Press.

[26] Jähne, B., (1995). *Spatio-Temporal Image Processing, Theory and Scientific Applications*, Vol. 751 of *Lecture Notes in Computer Science*. Berlin: Springer.

[27] Knutsson, H., (1989). Representing local structure using tensors. In *Proc. 6th Scand. Conf. Image Analysis*, pp. 244–251. Finland: Oulu.

[28] Carlson, S., (1994). The double algebra: an effective tool for computing invariants in computer vision. In *Applications of Invariance in Computer Vision*, J. L. Mundy, A. Zisserman, and D. Forsyth, eds., Lecture Notes in Computer Science, 825, pp. 145–164. Berlin: Springer.

[29] Hestenes, D., (1986). *New Foundations for Classical Mechanics*. Dordrecht: Kluwer Acad. Publ.

[30] Porteous, I. R., (1995). *Clifford Algebra and Classical Groups.* Cambridge: Cambridge University Press.

[31] Yaglom, I. M., (1968). *Complex Numbers in Geometry.* New York: Academic Press.

[32] Blaschke, W., (1960). *Kinematik und Quaternion.* Berlin: Deutscher Verlag der Wissenschaften.

[33] McCarthy, J. M., (1990). *Introduction to Theoretical Kinematics.* Cambridge, MA: MIT Press.

[34] Murray, R. M., Li, Z., and Sastry, S. S., (1994). *A Mathematical Introduction to Robot Manipulation.* Boca Raton: CRC Press.

[35] Tweed, D., Cadera, W., and Vilis, T., (1990). Computing tree-dimensional eye position with quaternions and eye velocity from search coil signals. *Vision Research*, **30**:97–110.

[36] Hestenes, D., (1994). Invariant body kinematics: I. Saccadic and compensatory eye movements. *Neural Networks*, **7**:65–77.

[37] Hestenes, D., (1994). Invariant body kinematics: II. Reaching and neurogeometry. *Neural Networks*, **7**:79–88.

[38] Hestenes, D. and Ziegler, R., (1991). Projective geometry with Clifford algebra. *Acta Applicandae Mathematicae*, **23**:25–63.

[39] Hestenes, D., (1991). The design of linear algebra and geometry. *Acta Applicandae Mathematica*, **23**:65–93.

[40] Lasenby, J., Bayro-Corrochano, E., Lasenby, A., and Sommer, G., (1996). A new methodology for computing invariants in computer vision. In *Proc. 13th Int. Conf. on Pattern Recognition*, Vol. A, pp. 393–397. Vienna: IEEE Computer Soc. Press.

[41] Kantor, I. L. and Solodovnikov, A. S., (1989). *Hypercomplex Numbers.* New-York: Springer.

[42] Rooney, J., (1978). A comparison of representations of general spatial screw displacements. *Environment and Planning*, **B,5**:45–88.

[43] Bayro-Corrochano, E., Lasenby, J., and Sommer, G., (1996). Geometric algebra: a framework for computing point and line correspondences and projective structure using *n* uncalibrated cameras. In *Proc. 13th Int. Conf. on Pattern Recognition*, Vol. A, pp. 333–338. Vienna: IEEE Computer Soc. Press.

[44] Funda, J. and Paul, R. P., (1990). A computational analysis of screw transformations in robotics. *IEEE Trans. Rob. & Automation*, **6**:348–356.

[45] Rooney, J., (1977). A survey of representations of spatial rotation about a fixed point. *Environment and Planning*, **B,4**:185–210.

[46] Aspragathos, N. and Dimitros, J., (1998). A comparative study of three methods for robot kinematics. *IEEE Trans. Syst., Man, and Cybern.—Part 13*, **28**:135–145.

[47] Daniilidis, K. and Bayro-Corrochano, E., (1996). The dual-quaternion approach to hand–eye calibration. In *Proc. 13th Int. Conf. on Pattern Recognition*, Vol. A, pp. 318–322. Vienna: IEEE Computer Soc. Press.

[48] Bregler, C. and Malik, J., (1997). *Video Motion Capture*. Tech. Rpt. UCB//CSD–97, 973, Berkeley: Comp. Sci. Div., Univ. of California.

[49] Clifford, W. K., (1873). Preliminary sketch of bi-quaternions. *Proc. London Math. Soc.*, **4**:381–395.

[50] Bayro-Corrochano, E., Daniilidis, K., and Sommer, G., (1997). Hand–eye calibration in terms of motion of lines using geometric algebra. In *Lappeenranta: Proc. 10th Scand. Conf. on Image Analysis*, pp. 397–404.

[51] Doran, C., (1994). *Geometric Algebra and its Application to Mathematical Physics*. PhD thesis, Univ. of Cambridge.

[52] Doran, C., Hestenes, D., Sommen, F., and Acker, N. V., (1993). Lie groups as spin groups. *J. Math. Phys.*, **34**:3642–3669.

[53] Ramponi, G., (1990). Bi-impulse response design of isotropic quadratic filters. In *Proc. IEEE*, 78, pp. 665–677.

[54] Schetzen, M., (1981). Nonlinear system modeling based on the Wiener theory. *Proc. IEEE*, pp. 1557–1573.

[55] Sicuranza, G. L., (1992). Quadratic filters for signal processing. *Proc. IEEE*, pp. 1263–1285.

[56] Zetzsche, C. and Barth, E., (1990). Fundamental limits of linear filters in the visual processing of two–dimensional signals. *Vision Research*, **30**: 1111–1117.

[57] Krieger, G. and Zetzsche, C., (1996). Nonlinear image operators for the evaluation of local intrinsic dimensionality. *IEEE Trans. Image Process.*, **5**:1026–1042.

[58] Bülow, T. and Sommer, G., (1997). Algebraically extended representations of multidimensional signals. In *Proc. 10th Scand. Conf. on Image Analysis*, pp. 559–566. Lappeenranta.

[59] Bülow, T. and Sommer, G., (1997). Das Konzept einer zweidimensionalen Phase unter Verwendung einer algebraisch erweiterten Signalrepräsentation. In *Mustererkennung 1997*, E. Paulus and B. F. Wahl, eds., pp. 351–358.

[60] Bülow, T. and Sommer, G., (1998). A new approach to the 2–D analytic signal. In *DAGM'98*. submitted.

[61] Hahn, S. L., (1992). Multidimensional complex signals with single-orthant spectra. *Proc. IEEE*, pp. 1287–1300.

[62] Hahn, S. L., (1996). *Hilbert Transforms in Signal Processing*. Boston: Artech House.

[63] Bülow, T. and Sommer, G., (1997). Multi-dimensional signal processing using an algebraically extended signal representation. In *Algebraic Frames for the Perception-Action Cycle*, G. Sommer and J. J. Koenderink, eds., Lecture Notes in Computer Science, 1315, pp. 148–163. Berlin, Heidelberg: Springer.

[64] Bayro-Corrochano, E. and Buchholz, S., (1997). Geometric neural networks. In *Algebraic Frames of the Perception–Action Cycle*, G. Sommer and J. J. Koenderink, eds., Lecture Notes in Computer Science, 1315, pp. 3l79–394. Berlin, Heidelberg: Springer.

[65] Buchholz, S., (1997). *Algebraische Einbettung Neuronaler Netze*. Diploma thesis, Cognitive Systems Group, Inst. of Comp. Sci., Univ. of Kiel, Germany.

[66] Arena, P., Fortuna, L., Muscato, G., and Xibilia, M. G., (1996). Multilayer perceptrons to approximate quaternion valued functions. *Neural Networks*, **9**:1–8.

[67] Georgiou, G. and Koutsougeras, C., (1992). Complex domain backpropagation. *IEEE Trans. Circ. and Syst. II*, **39**:330–334.

[68] Nitta, T., (1997). An extension of the back-propagation algorithm to complex numbers. *Neural Networks*, **10**:1391–1415.

[69] Pearson, J. K., (1994). *Clifford Networks*. PhD thesis, Univ. of Kent.

[70] Bayro-Corrochano, E., Buchholz, S., and Sommer, G., (1996). Selforganizing Clifford neural network using geometric algebra. In *Proc. Int. Conf. Neural Networks, Washington, DC*, pp. 120–125.

Part II

Industrial and Technical Applications

11 Market and Future Needs of Industrial Imaging

Klaus Singer

F1 professional services GmbH, Ginsheim-Gustavsburg, Germany

11.1 Introduction

In this chapter of the book, the actual market situation of industrial imaging will be shown. ‘Industrial’ is very important. We are concentrating on industrial applications of the imaging technique, which can be divided in the following general areas: Surface control, position detection, identification, completeness control, and optical measurement.

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

This chapter does not deal with applications in the wide fields of multimedia, document processing and archiving, image communication, medical applications of imaging techniques, or video surveillance.

Today's industrial imaging market is relatively small compared with these branches. One reason for this status may be the fact that industrial imaging has to fulfill the difficult task of interpreting images. Multimedia and the other fields of imaging mentioned here mostly handle images as images in order to transport, display, store and manipulate them. Mostly, they stay at an image level.

Industrial imaging nearly always has to draw conclusions from images. For instance, an object must be identified, positions have to be detected, codes have to be read etc. Thus, for industrial imaging the image is only a starting point. The objective is to reduce an image, which can consist of millions of pixels, to only a few bytes indicating some properties of an object. As an extreme, only one bit has to be generated, which then means 'part is good' or 'part is bad'. We all know how difficult this task can be. Therefore, it is understandable that the industrial imaging market is currently small. As we will see, however, the future vision is exciting, especially if the industrial vision industry is able to create standard products that can supersede today's engineering orientation.

The German market figures presented thereafter are mainly taken from annual market reviews as they have been carried out by the German VDMA e.V. (*Verein Deutscher Maschinen- und Anlagenbau*, [1]). International market data were taken from other surveys as they were published, for instance by the *AIA* (*Automated Imaging Association*, [2]). The VDMA is the oldest and one of the most important German industrial organizations. Its members are mainly machine and production plant building companies. In late 1994, the organization decided to establish a new working group under its rubric, which would concentrate on industrial imaging. This chapter of the book deals mainly with the German market of industrial imaging. Its status seems to be similar to other developed national markets. Therefore, it is believed that the conclusions drawn are valid not only for this country, but can be regarded as a general trend. Nevertheless, relevant international market data are given to show the worldwide context.

11.2 Historical roots

11.2.1 Early applications

The first significant image processing applications of which the author is familiar are approximately thirty years old. Satellite imaging, weather observation, and scientific tasks were the starting points. Most of these applications have been driven by military requirements.

At that time, image processing was a very expensive and, therefore, very exclusive technique. Only a few systems were counted. The status more or less remained at this level until the mid-1980s.

With new technologies on one hand and new demands on the other, the situation changed from year to year. Today we find a relatively small but constantly growing market that is far more important for its customer branches than the pure market data have led us to believe.

11.2.2 System architectures

Early image processing systems were not systems in the sense we see them today at our desk or in production plants. Their architecture was very special and dedicated to a specific purpose. Most of the algorithms were implemented in pure hardware. That is the reason why these systems could be considered the graveyard of digital logic circuitry. Thousands of members of the formerly well-known 74xxx TTL-family could be found on hundreds of huge printed-circuit boards in many tall electronic housings. Such a system had a power consumption of 100,000 W or even more. A simple workroom was inadequate; a ballroom-sized one was necessary.

The hardware architecture most often used was of pipeline structure. Most image processing was performed in video realtime.

Other semiconductor technologies like *emitter coupled logic* (ECL) were developed to speed up processing while reducing power dissipation. With the upcoming digital signal processors (bit slice processors and others) and semiconductor memory devices, the implementation of imaging systems became much easier. Other architectures occurred, but it was still a very hardware-oriented approach. And each vendor of imaging systems defined his own standard as to how the different modules of the system had to be interconnected.

Within a certain period of time between 1985 and 1990, the *VME bus* played an important role in industrial imaging. This bus concept was very well accepted because of its bandwidth, its ruggedness, and a certain degree of freedom, allowing the implementation of special dedicated image processing architectures and sub-buses.

Later on, several digital video buses were proposed to create modular systems. In Germany, the Imagestar project with significant public funding worked on this issue in the late 1980s. In 1993, the German *Vision Club*—now part of the *VDMA*—introduced *digital video link* (DVL), a simple interface method for both attaching digital cameras and for intrasystem communication. The basis for this approach was a high-speed bit-sequential link using so-called TAXI devices from *advanced micro devices* (AMD). From this general approach it can be compared with newly proposed standards (FireWire etc.).

These trials failed as long as the intra-system communication was involved. They had a small success in the field of camera connection, but only small camera vendors did support these efforts. Larger vendors preferred to do nothing, establish their own standard or use existing analog video interfacing.

In the author's opinion, the main reason why all these efforts concerning video-bus standardization failed was the dramatic development of general-purpose processors: complex vision systems and special buses were no longer necessary.

11.2.3 Sensor technology

The very first sensors used with imaging systems were based on vacuum tube technology. One can imagine that this technology was difficult to handle. The mechanical form-factor gave further limitations. Sensor linearity was bad. Another sensor technology was based on diodes or diode-arrays.

It is evident that this sensor technology was as limited as the system technology was.

About 25 years ago, CCD technology arose, driven mainly by consumer and video surveillance applications (CCTV). Today, CCD sensors are the most used type in industrial imaging. The technology provides high-resolution sensors. Using mosaic filters or three-sensor set-up, color images can be generated.

Most cameras used in industrial imaging are still matrix cameras working with the interlaced video format. This format had been created to offer a reasonably good compromise between bandwidth and resolution in TV broadcasting systems. This standard might have fulfilled this objective.

For industrial vision applications, this standard is possibly one of the worst approaches. Normally, objects in a production environment are not static but moving. Using TV-interlaced technique, two sequential image fields are generated showing a moving object at two different locations.

It is amazing that this standard became the video format most often used in industrial imaging as well. The reason for that might be the limited bandwidth on the digitization level, the same reason for it in TV broadcasting. The analog video signal has to be digitized prior to processing. At an earlier stage of semiconductor development, this was difficult to implement in standard technology. And this is still a slight limitation: buses limit transfer rates as today's standard memory architectures do.

The *CCD technology* has been proven, but some severe limitations are associated with it. There is, for instance, its limited dynamic range. Best case CCD sensors have a range of three decades. Therefore, imaging applications with CCD cameras are very sensitive against variation in illumination. For example, sudden sunlight can lead to severe image disturbances. On-line tracking of a laser-based welding process is an interesting task for industrial imaging, but its realization is very difficult based on CCD technology. Another example is video surveillance. With a CCD camera, it is nearly impossible to observe a scene with mixed indoor and outdoor light with the expectation of getting usable results in both image areas.

To summarize, CCD technology has reached a high level in image quality and it is light sensitive, but due to the internal architecture the number of frames per second is limited, image information is available only on a line-by-line basis, the dynamic range is small, and the production costs are relatively high. This is especially true for production costs of cameras. A CCD sensor needs some external control circuitry and it is not possible to integrate it on the sensor die.

Promising new efforts have been on the way in the last several years. The key seems to be CMOS technology (Chapter 26; Volume 1, Chapters 7 and 8).

11.3 Market overview

Before discussing German market data in detail, an international context will be given. The North American market reached an overall 1.5 billion U.S. dollars by the end of 1997. Equivalent market size for Europe (including Israel) is seen at slightly more than 950 Mio U.S. dollars, compared to about 820 Mio U.S. dollars for 1996. Excluding value added by system integrators and OEM-type companies, the total revenue was about 1230 billion U.S. dollars for North America and 710 Mio U.S. dollars for Europe. The world-wide market for machine vision now has nearly reached 4 billion U.S. dollars [2].

The North American market has grown by a factor of nearly 3 during the past five years. For the next 5-year period a similar growth trend is forecasted. The European vision market will grow at a rate of about 15 % in the next several years. Due to falling prices, the growth rate in units is not far from twice the revenue growth rate. Unit prices will continue to fall. On this international data basis we will now analyze the German market in detail. As said earlier, we believe that the results are representative for the world-wide machine vision market. Special properties of the German market—if any—are pointed out.

11.4 Economical situation

The following market figures were obtained from a market survey by the end of 1997 [1]. It was organized by the VDMA for the third year in

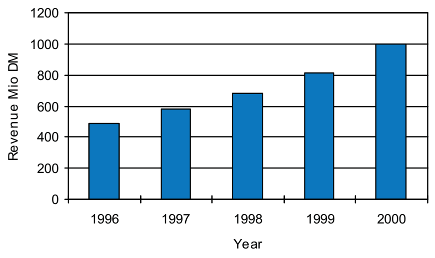


Figure 11.1: *Volume of the German machine vision market from 1996 with forecasts until 2000. (Revenue in Deutschmarks (DM).)*

a row. A similar investigation was done by the Vision Club e.V. for the years 1992, 1993, and 1994.

The questionnaire was sent to about 200 German companies and institutions, all of which are active in industrial imaging. The data base is more or less identical with the one used in 1996. In 1997, about 33 % answered, whereas in 1996 the reflow rate was 20 %. Even 20 % seems to be a good result compared to experiences with other surveys. The 33 % for 1997, therefore, gives quite a solid basis.

11.4.1 Market volume

The total market volume by the end of 1997 reached 580 Mio DM with a growth rate of 16 % compared to 1996 (Fig. 11.1). The expectations are 18 % for 1998, nearly 20 % for 1999 and more than 20 % for the year 2000. All past market reviews always lead to the result that expectations and real growth were at a level of about 15 % on a year-to-year basis.

Compared to total Europe, the German market seems to be the largest market in industrial imaging. Worldwide, the German market is in one of the top positions and the U.S. market is, of course, the largest one.

11.4.2 Structure of suppliers

Using the latest market reviews, a clear trend is to be seen concerning the structure of the supplier side of the market.

On one hand, it is true that smaller companies dominate. Sixty percent of these companies have fewer than 50 employees. Within the class of 10 to 19 employees, the number of companies increased by about 8 % from 1996 to 1997, where the number of companies with 5 to 9 employees decreased by about 14 %. The number of companies with 5 or fewer employees increased from 24 % in 1996 to 30 % in 1997. In 1996, about 5 % of the companies had more than 50 employees. In

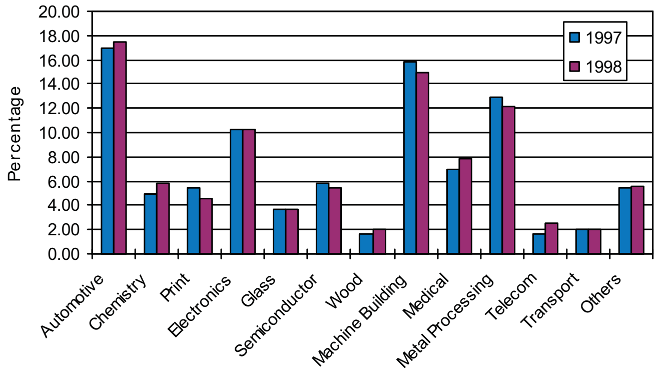


Figure 11.2: *Types of the companies involved in the German machine vision market.*

1997, more than 9 % were found in this class. Based on the fact that the total number of vendors in the market stayed stable during recent years, the conclusion is that the average company size is growing.

Industrial imaging is a relatively young branch. About 2/3 of the companies, especially the imaging departments within larger enterprises, have been active for 10 or more years. The average company age increases from year to year.

The distribution of the turnover per company is constantly shifting to higher volumes. For 1998, it is expected that the partition of annual revenues up to 3 million DM will drop below 50 %.

These facts allow the conclusion to be drawn that industrial imaging suppliers are very well established. They have a solid basis of knowledge and expertise. They are small enough to act flexibly and rapidly, fulfilling the demands of their customers.

About 45 % of the companies regard themselves as system houses (Fig. 11.2). Twenty percent are integrators, and OEM manufacturers are at a level of about 15 %. The rest is divided into Value Added Reseller (VAR) and others (consultants, etc.). This structure is stable compared to results in the past.

The export rate is about 25 % as far as direct transfer is considered. North America is the leading export region with nearly 12 % of all exports. The expectation for 1998 is a growth to nearly 13 %. Switzerland follows with 10 % and a slight increase for 1998. Asia contributes nearly 9 %, with a negative growth expected for 1998.

As far as indirect transfer is considered, about another 33 % is estimated to be exported through German industrial imaging customers.

The high export rate for mainly engineering oriented goods can be taken as another proof of the high quality and reliability of industrial

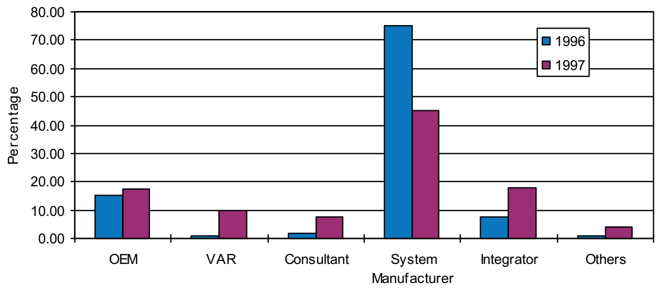


Figure 11.3: *Main fields of application for the machine vision market.*

vision systems. No customer takes the risk of shipping a system around the world if he does not trust in it.

11.4.3 Main fields of applications

First of all, why do customers decide to use industrial imaging systems in their production plant? About 2/3 answered that *quality control* (quality assurance) was the reason. The weighted result is 35 %. The next important reason for using imaging is *production automation*. About 1/2 of all answers were based on this. The weighted percentage is 25 %. Both reasons for installation of industrial imaging systems will add to future growth figures (Fig. 11.3).

These results follow a past trend. Three reasons are important and should provide sufficient explanation.

First, globalization demands a very high quality level for goods on the world market. It is far more expensive to service defective deliveries over long distances than to establish highly sophisticated quality control strategies. And, of course, globalization leads to stronger competition with the effect that prices decrease while increasing quality is requested. The second reason is related to legal issues. More and more the producer will be made responsible for damages caused by his product. Third, growing automation itself needs quality control, not only at the end of the production pipeline, but more and more between relevant production steps. In the past, there were more human workers involved in the production process than is the case today and will be even more the case in the future. While handling goods, operating machinery, etc. they avoided defects implicitly. Today, the degree of automation has reached such a level that even a simple failure can cause significant damage and production interruption. Because of the just-in-time production strategy, every unexpected distortion can have a domino effect with dramatic costs.

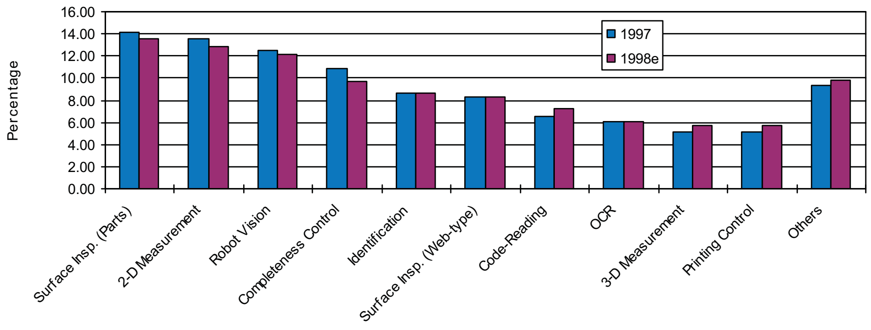


Figure 11.4: *Customer branches for the machine vision market.*

Image processing is one of the few methods by which quality can be controlled in a nondestructive way with always the same parameters and an automatic documentation if desired.

The general tasks to be fulfilled by industrial imaging are *surface inspection* of parts with a share of about 14 %, *2-D measurement* with about 13 %, *robot vision* (position detection) with just more than 12 %, and *completeness control* with about 10 %. The importance of all these tasks is expected to decrease slightly in 1998. At the growing edge, *identification by code reading* is located with a plus of 1 % to 6.5 % for 1997 and *3-D measurement* plus nearly 1 % added to 4.8 % in 1997.

11.4.4 Customer branches

Far more than 20 different answers were given to the question, to which branch does the customer belong? Again, from another aspect this shows the huge potential and versatility of industrial imaging.

Compared to 1996, the *automotive industry* now takes first place. Nearly 17 % saw future growth potential. The former number one, *machine builders*, are now number 2 with nearly 16 %, losing another expected 1 % in 1998. The *metal processing industry* came to 14 % with a tendency down to 13 % in 1998. The *electronic industry* is number four at slightly above 10 % and is expected to remain stable at this level in 1998. The *pharmaceutical industry* comes next with 7 % for 1997 and nearly 8 % for 1998 (Fig. 11.4).

11.4.5 Products versus engineering

The structure of the vendors' side of the market already shows a strong trend to engineering orientation. More than 80 %, with a slightly decreasing trend in 1998, answer that their companies deliver complete systems and services. The rest of the market involves components.

Software tools, computer parts, cameras, lighting, and optics are more or less on an equal level with a slight growing tendency for software tools.

Today, industrial imaging in Germany is dominated by complete, *application-specific solutions*. Driven by a strong need and supported by a large number of successful installations, the market is very stable and constantly growing.

In long-term consideration, however, the engineering orientation could lead to a self-chosen limitation. Obviously, in general, good engineers are rare. Especially in the industrial imaging industry, very high requirements have to be fulfilled. The vision engineer is a real all-round expert. On one hand, he must be a software expert, on the other hand, he has to have very high-level knowledge of signal processing and industrial control engineering.

Real growth potential is possible if the vision industry moves towards areas in which these types of products are more and more important.

At this point, some U.S. vision companies are a step ahead. They have emphasized the development of products which can be handled by the end-customer himself or third-party integrators (VAR-type companies). It is no accident that the main end-user branch these companies are targeting is the electronic industry. From the sight of an industrial vision vendor, this is a field with many similar, highly automated production steps. Production volume is very high. The environmental conditions concerning handling and lighting are stable. Together with these companies, the U.S. market differs somewhat from the German and other national markets.

11.5 Mainstream technology used today

As discussed in tandem with the historical roots of industrial imaging, the dramatic role of *general-purpose microprocessor* development was pointed out.

If we look today at a typical vision system for industrial use, we find no special hardware architecture. Microprocessors are the basis for most of the processing and so-called frame grabbers provide the digital image data in a certain system memory area using DMA-technique. Sometimes, one finds special hardware doing some filtering in the video data flow. However, this is relatively rare, and this is in fact the reason why more and more semiconductor vendors discontinue their production of such dedicated video devices.

Regarding the bus standard used inside, the system PCI is the keyword today. Because of existing PCI bridges, it is quite easy to interface cameras to a PCI-system. Video A/D-converters with integrated

DMA circuitry and on-chip PCI interface allow us to offer cheap add-on frame-grabber boards.

General-purpose microprocessors have reached a computational power of 200 million instructions/s and more. Certain processor families even have built-in execution units that are dedicated to perform certain basic algorithmic kernels often used for image processing very efficiently (Chapter 3). The force behind this development was not the *industrial imaging industry*. The vision industry benefits from *multimedia* and other areas.

Image processing tasks are normally very complex. Therefore, a good development environment is needed to achieve short *time-to-market*. Because industrial vision deals with images, a state-of-the-art user interface of the computer is needed for the presentation of not only text, but also of graphical and pixel data. And last, but not least, image data need extensive hard-disk storage space. All these are technical reasons for why today's vision system is based mostly on a standard PC architecture.

By using standard PC systems sometimes even video realtime performance can be reached, which is by far not required in most industrial imaging applications. Thus, PC systems normally have enough reserves in computational power to achieve production realtime using sure and stable algorithms.

In applications like constant surface inspection of endless material, of course, video realtime processing power is necessary. This is one of the last fields remaining in industrial imaging, where a dedicated system architecture and/or special processing hardware could be useful.

11.6 Future trends

11.6.1 Future sensor technology

The severe disadvantages of current CCD sensor devices were already discussed. In the near future some very important changes will occur.

The human eye has a dynamic range of about 6 decades. This is a target for further development of sensor technology. Over the last few years, new promising efforts have been made to use CMOS-technology for the next generation image sensors. Currently, different concepts like APS (Volume 1, Chapter 7), HDRC (Volume 1, Chapter 8) or others are presented with reasonably good results. Still some technology problems must be solved. For instance, image quality has to be improved. Fixed-pattern noise is the reason for the apparently lower image quality of today's sample devices. But, most probably, *CMOS sensors* with a dynamic range of a minimum of five decades with sufficient geometric resolution will go into mass production within the next 15 mo.

The main advantages of CMOS sensors are lower production costs, lower power consumption, and higher dynamic range. For some applications it might be useful to access pixel array randomly. The frame rate is no longer limited to more or less TV-standard. One of the most interesting features is the possibility of integrating the complete control logic on the sensor die. Even A/D converters or some on-line filter functionality could be implemented. Complete subsystems could be available in the near future on a single sensor chip.

Imaging in total will benefit dramatically from this new sensor technology, because it leads to cheaper cameras with easier handling and cheaper lenses. This will be a very important step in creating new generations of industrial imaging systems. The special properties of CMOS sensors allow implementing much more robust image processing algorithms. The systems in total will climb to a new level of robustness. Higher quality at lower prices will become available.

Again, the development in sensor technology is not driven by industrial imaging. Market estimations say that in the year 2000 the worldwide imager market will have a volume of far more than 1 billion U.S. dollars. Costs per piece will drop down to a level of 10 U.S. dollars. Most image sensors will be used in multimedia type applications, video surveillance, and video communication.

11.6.2 Future imaging system technology

In the near future industrial imaging will continue to benefit from the development that is driven by multimedia and other related applications. For instance, a large portion of the 65 million PCs in the year 2000 will already be equipped with digital cameras. It is sure that some major improvements in standard PC architecture will take place to attach digital cameras much more easily than is now possible. In this context, some proposals (*FireWire* etc.) are currently under discussion. Very soon, such an interface could belong to a standard PC system as a serial communication line does nowadays.

The computational power will grow by a factor of about 2 every year. Microprocessors with 64-bit architecture will become the next generation standard. Costs for RAM-devices and mass storage will further decline. As a result, system costs will decrease while system power and functionality will increase.

Of course, some time lag will occur between trend-setting for standard PC systems and its penetration as industrialized components. As there was a parallelism between them in the past so it will be in the future.

The real disadvantage of such a system architecture becomes more important as industrial vision applications move from *application-specific systems* to *standardized products*.

On one hand, space is a problem. The PC-based imaging systems need a rack to be mounted. They normally need a monitor and a keyboard, and other space-consuming components. Power consumption requires additional space and certain provisions for air convection. Second, most used disk operating systems require a definite power-on and power-off procedure. Third, cabling between the remote camera and the system is sensitive concerning electric noise both in a passive and active sense. All these facts result in visible and invisible costs and lower the level of acceptance.

The next large step on the way to further integration and miniaturization is the concept of an *intelligent sensor*. Today's semiconductor integration allows a very comprehensive design of an all-in-one image processing system. In a small housing the sensor, the processing unit, and the I/O-level are integrated. Power consumption is one to two orders of magnitude lower than with an PC-system architecture. No sensitive video cabling exists, because video processing is done directly behind the sensor. The intelligent camera needs no hard disk drive. Program and data are stored in EPROM-devices. No fan is necessary and no other moving or turning parts are needed. The computational power might not reach PC level, but it will be high enough to perform many standard applications. Using modern embedded controllers, programming on assembler language level as with traditional *digital signal processors* (DSPs) is no longer necessary. This results not only in shorter time-to-market, but is also a condition for high quality, high reliability and maintainability.

The costs of such a solution are significantly below a PC-based vision system, especially if all the invisible costs are taken into account.

Such an *intelligent camera* with preconfigured dedicated functionality could be easily handled by customers without specific image processing knowledge. Acceptance regarding imaging technology will increase and products and services of industrial imaging will become more and more in demand.

11.7 Conclusions

At this point we can draw some conclusions. We have explored the status of the industrial imaging industry, its economical situation, and technical background concerning system architecture and sensor technology.

We should learn from the past and raise the question: which conditions must be fulfilled in the future to guarantee further significant development of this industry branch?

At the end of this chapter four theses will be developed to concentrate these results.

11.7.1 The role of industrial imaging

We have seen that the industrial imaging market is still relatively small, but steadily growing. The vision industry is very well established, and the suppliers have solid structures and great expertise. Industrial users of vision systems trust in this technology and regard them very often as the best or even the only way to lower costs and increase quality. These are the conclusions of the recent VDMA market survey [1]. Continuity of the results over the last years is as well a proof of this.

The question is how large are the advantages for customers installing and using vision systems in their production?

In the market survey for 1997 the VDMA tried to get a first rough number of this issue. The equivalent item in the questionnaire submitted to the industrial vision suppliers was: Please estimate the ratio between your customer's revenue with products using vision technique and your turn-over regarding industrial imaging?

Thirty-eight companies gave an answer with a factor of 2 at the lower end and 300 as the highest number. The resulting average is 38. Therefore, products and services of industrial imaging are influencing about 22 billion DM. This corresponds to about 10 % of the total revenue of the German machine and production plant building industry.

It is evident that this number has to be handled with care. On the other hand it is not very important, whether we talk about 15 or 30 billion DM. Important is the message itself, that there is quite a significant multiplication factor.

Thesis 1 *The industrial vision industry will continue to play a central role in maintaining and increasing profitability and competitiveness because of the worldwide trends in increasing production automation and globalization and due to legal issues.*

11.7.2 Future needs of the imaging industry

Obviously, industrial image processing requires very wide and deep knowledge in signal processing, optics, process control engineering, and software design. It is evident that skilled engineers with an excellent education are one key to further development of this branch of industry. Excellent education presupposes an excellent system of schools, universities, and excellent teachers. Excellent contacts and cooperation between universities and technical colleges must take place.

In Europe these conditions are currently in place. But will we get more and still better engineers in the future? In which direction future development will go remains an unanswered question.

Thesis 2 *Industrial image processing needs excellent engineering.*

As we have seen, industrial image processing is quite a complex task. To draw conclusions out of images is often difficult. Excellent engineering potential alone does not help. Sophisticated software tools and development support are necessary to achieve short time to market and create stable and secure solutions.

What is meant regarding software tools in this context? On one hand the vision industry needs good operating systems with compilers (C, C++, etc.), debuggers, and so on. This is more or less understood. No one today can afford to develop sophisticated software in assembly language; indeed, software in that sense is on the market today.

But software tools must be understood in a different sense as well. What the industry needs are sophisticated development systems that provide basic algorithms and more.

Today, nearly every developed imaging supplier has created his own and special tool package. Inventing the wheel a second time still continues and engineering power is wasted. The reason for this status is evident. Every supplier tries to keep his “secrets.” And, of course, it is difficult to break with the past and switch to another development platform. On the other hand, the real “secrets” of an industrial vision supplier are surely not the way to implement a filter or everything else. The real know-how involves solving a certain task by using and combining such basic image processing algorithms.

We have seen the development from hardware- to software-oriented systems. But on the tool level we are still years behind.

Today, it is a dream to have a software tool that supports applications in industrial imaging on several levels. First, the tool provides a rich collection of basic image processing algorithms. Second, the tool supports the creation of higher level macrotype functions, which can be handled as modular components. Third, a configuration module allows for control of the data flow and combines selected macrotype functions to an executable program. Fourth, an I/O module cares about coupling the image processing system with the outer world, where “outer world” means on-line process environment and user-interaction mainly during teach-in time. Fifth, different target generators support different system architectures, like PC-based systems on one hand and embedded systems (intelligent cameras) on the other.

Actually, there are already some powerful tools on the market and new efforts were undertaken recently to create a standard environment. Existing products cover different levels of functionality from basic libraries up to application-oriented packages (optical character reading, etc.). But they all represent only certain selections and have important limitations. The dream still waits to be fulfilled ... .

Thesis 3 *Industrial image processing needs very powerful, flexible, robust systems (and sensors) with very good development support.*

As we saw, industrial imaging is highly engineering-oriented. An image processing application today is driven mainly by the customer's special wishes. Very often the customers request a complete system, consisting not only of the image processing system and the sensor part, but also with specialized mechanical handling and so on. Resulting lot sizes are low and it is not rare that they drop to one.

Engineering is a limited source. It works fine especially in relatively small organizations, where it can be very productive. Engineering services cannot be multiplied as easily as products can be. So the strength of today can turn into a limitation of tomorrow. As long as the customers of industrial imaging insist on a fully customized solution, products and services will be expensive, which results in another limitation on growth. But the more customers and suppliers look for good compromises in order to get more standardized products and services, the less expensive they can be and the more they will be used. This is, of course, a long-term process.

Intelligent cameras can play an important role. They provide an excellent price-performance ratio. They are compact, have easy-to-understand interfaces, both on hardware and software level. Based on state-of-the-art sensors, robust solutions are possible for a huge variety of standard tasks. Normally, intelligent cameras come with a PC-based configuration software, which enables even an end-user with no special image processing knowledge to edit inspection tasks interactively. The PC is only necessary in the configuration phase. In production mode, the device works stand-alone. Currently, target application areas for such solutions are mainly completeness control, 2-D measurement, and code reading.

Industrial imaging suppliers should make a tremendous effort to create products that can be regarded and handled as simple components for automation. In this context the concept of intelligent cameras will play a key role.

Thesis 4 *The industrial vision industry will only be able to fulfill existing (as opposed to strictly theoretical) future demands if more and more standardized products enter the market.*

11.8 References

[1] VDMA e.V., (1998). Marktbefragung Industrielle Bildverarbeitung/Machine Vision 1997/1998. VDMA e.V., Frankfurt, Germany.

[2] Automated Imaging Association (AIA), (1998). The machine vision market: 1997 results and forecasts through 2002. Ann Arbor, MI, USA.

12 Applications of Morphological Operators

Pierre Soille

Silsoe Research Institute, Silsoe, Bedfordshire, United Kingdom

12.1 Introduction

This chapter[1] provides a brief overview of published papers reporting on applications that have been tackled with morphological operators. The papers are sorted by application fields (Sections 12.2–12.8): geosciences, material sciences, biomedical imaging, industrial applications, identification and security control, document processing, and image coding. Topics not fitting into these categories are discussed in Section 12.9. The list of references given at the end of this chapter is not meant to be comprehensive. In particular, only references published in image processing journals are cited (with a few exceptions due to the author's own research interests).

[1]A preliminary version of this chapter appeared in [1, Chapter 11].

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

12.2 Geosciences

The morphological extraction of linear structures in satellite images of the earth surface is studied in Destival [2]. The design of an adaptive morphological filter suited to the removal of stripes in satellite images is developed in Banon and Barrera [3]. The use of *alternating sequential filters* for *speckle removal* on *radar images* is detailed in Safa and Flouzat [4]; Martel et al. [5] have shown that morphology can be regarded as a good alternative to harmonic analysis for determining characteristic features of a function defined on the earth's surface. This is shown for the detection of *gravity anomalies* using directional morphological filters. The morphological processing of infrared ocean images for extracting areas of closed circulation (vortices) using opening and closing operators is introduced in Lea and Lybanon [6]. Watershed-based clustering applied to the *classification* of *satellite images* is discussed in Watson [7], Watson et al. [8], and Soille [9]. An example of a classification result is shown in Fig. 12.1.

Applications dealing with the analysis of topographic data are detailed in Chapter 19.

Zheng et al. [10] show that granulometries with line segments and disk-shaped structuring elements allow the discrimination of tree species from aerial photographs.

The marker-controlled segmentation is applied to the analysis of images mapping the electrical conductivity along boreholes in Rivest et al. [11].

12.3 Material sciences

An overview of some morphological image analysis techniques pertinent to materials sciences is presented in Berman et al. [12]. Shape analysis using granulometries, the quench function of the skeleton, and parameters derived from the propagation function are detailed in Chermant et al. [13] and applied to the characterization of graphite, nickel, and silicon particles. Many applications of morphology to material sciences are also described in Coster and Chermant [14].

Propagation algorithms for predicting macroscopic properties of heterogeneous media are proposed by Jeulin et al. [15]. The main idea consists in replacing the continuous medium by a restricted set of relevant nodes and then applying morphological operators on the corresponding adjacency graph. In Demarty et al. [16], the contact permeability between rough surfaces acquired with a *confocal microscope* is estimated using 3-D geodesic propagation for detecting the minimal paths linking the flow source and sink.

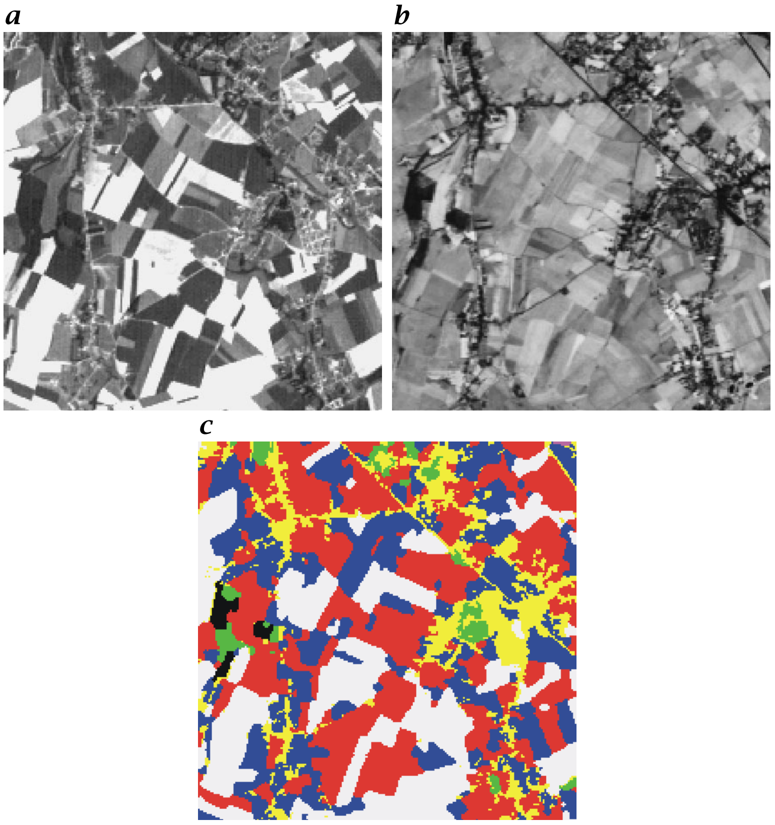


Figure 12.1: Classification of a multispectral satellite image (the two input bands are shown as grayscale images). The result of the classification is shown as a labeled image, a specific color being used for each ground cover (e. g., yellow for urban areas and green for forests). From Soille [9]; (see also Plate 1).

Length and diameter estimations of man-made mineral fibers have been studied in Talbot [17] and Talbot et al. [18]. These measurements required the development of an efficient methodology for separating connected fibers in polished cross sections or crossing fibers in scanning electron microscope images. This is illustrated in Fig. 12.2.

The analysis of fibers in noisy and low contrast scanning electron microscope images proposed in Viero and Jeulin [20] is based on an algorithm operating on region adjacency graphs built from a watershed segmentation of the input image.

The *roundness* of grains such as sand particles is defined in Pirard [21] using distance transforms and granulometries. The use of morphological probabilistic models for characterizing mineral powders mixtures is investigated in Jeulin et al. [22].

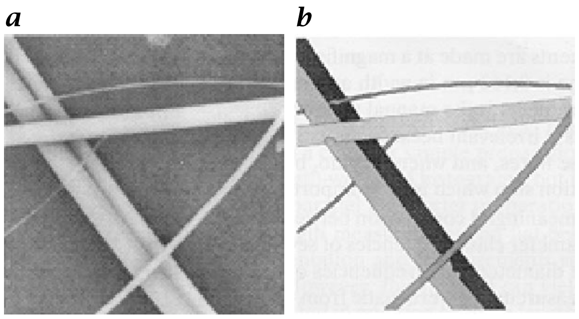


Figure 12.2: *Extraction and separation of fibers:* ***a*** *scanning electron microscope image of mineral fibers (backscattered electron mode);* ***b*** *resulting segmentation. Courtesy of Dr. H. Talbot, CSIRO Math. and Inf. Sciences, Sydney; see also Talbot [19].*

The detection of the boundaries of convex polytopes such as those encountered in 3-D images of *polyurethane foam* bubbles is detailed in Gratin and Meyer [23].

The journals *Journal of Microscopy*, *Microscopy, Microanalysis, Microstructures*, and *Acta Stereologica* regularly publish applications of mathematical morphology to material sciences.

12.4 Biological and medical imaging

The automatic screening of cervical smears using top-hat transformations and skeletons is proposed by Meyer [24]. The processing of 2-D *electrophoresis gels* using top-hats for normalizing the background and union of openings for removing the vertical and horizontal streaks is detailed in Skolnick [25].

Three-dimensional morphology is illustrated in Gratin [26] with many medical imaging applications such as the segmentation of magnetic resonance images of the brain and temporal image sequences of the heart. The analysis of confocal images of cells using geostatistical and morphological tools are introduced in Conan et al. [27].

The combination of the automatic watershed-based segmentation with user-defined markers is proposed in Higgins and Ojard [28] for better segmenting of 3-D cardiac images. The use of watersheds for segmenting echocardiographic images is also investigated in Zehetbauer and Meyer-Gruhl [29].

The fusion of images derived from magnetic resonance and x-ray computed tomography scanners using a morphological multiresolution image representation (pyramids) and decomposition technique is presented in Matsopoulos et al. [30] and Matsopoulos and Marshall [31]. The authors show that their approach performs better than linear

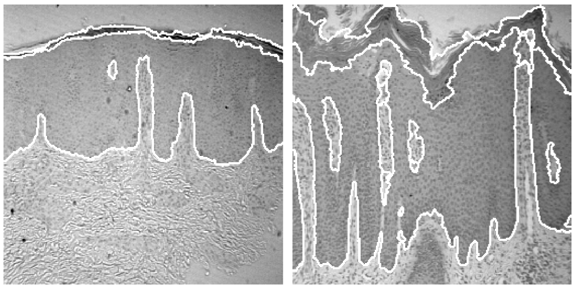


Figure 12.3: *Two skin biopsy images segmented using morphological filters and watersheds. Courtesy of Dr. J.R. Casas, Universitat Politecnica de Catalunya, Barcelona; see also Casas et al. [32].*

schemes in the sense that it preserves the local contrast (no blurring effect).

Casas et al. [32] developed a methodology for segmenting skin biopsy samples. It consists of a prefiltering stage based on alternating sequential filters by reconstruction followed by a marker-controlled segmentation. Two examples are shown in Fig. 12.3.

The detection of microcalcifications on high-resolution x-ray images of breasts (mammograms) is studied in Vachier [33].

The segmentation of corneal endothelial cells using a marker-controlled segmentation followed by measurements performed on the resulting region adjacency graph is detailed in Vincent and Masters [34]. The characterization of cork cells segmented by the watershed transformation is presented in Pina et al. [35]. The automatic recognition of cancerous cells using *morphological openings by reconstruction* and *ultimate eroded sets* is investigated in Thiran and Macq [36].

In Manders et al. [37], an alternative to the watershed transformation is presented and called the “Largest Contour Segmentation.” The algorithm consists in stopping the flooding of a catchment basin as soon as it meets another catchment basin. This procedure is applied to the detection of aggregated fluorescent microbeads and 3-D confocal images of biological materials.

12.5 Industrial applications

A prototype for an automated visual on-line metal strip inspection system is proposed in Laitinen et al. [38]. Morphology is used to suppress the uneven background illumination function and detect several defect types such as spills, scratches, and cracks.

The automatic detection of defects on x-ray images of engine parts is proposed in Jochems [39] and Jochems and Préjean-Lefèvre [40]. A

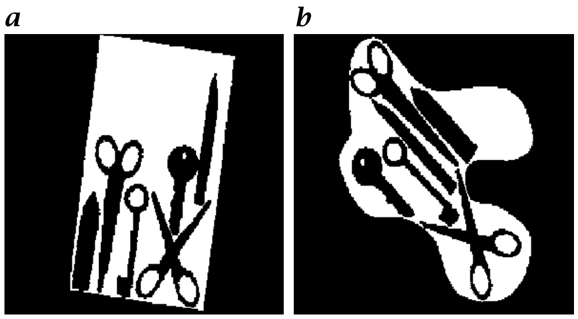


Figure 12.4: *Automatically packed items* ***a*** *into a regular tray; and* ***b*** *an irregular scene. Courtesy of Dr. P. Whelan, Dublin City University.*

study of the detection of defects in the textile industry is detailed in Müller and Nickolay [41].

The use of morphological operations for optimizing the packing of arbitrary 2-D shapes is discussed in Whelan and Batchelor [42, 43]. For example, Fig. 12.4 shows the results of packing some standard household items into a rectangular tray and an irregular scene.

A comparison between Fourier and morphological approaches for the on-line detection of watermarks in a paper factory is provided in Whelan and Soille [44].

The automatic extraction of the nodes of grid patterns painted on metal sheets to evaluate their deformation after stamping is solved using morphological directional filters and skeletons in Tuzikov et al. [45] and Peyrard et al. [46]. The unwrapping of interferometric phase maps using morphological gradients and watersheds for delineating the fringes is described in Soille [47].

The use of mathematical morphology for solving industrial image analysis problems is illustrated through several examples in Breen et al. [48].

12.6 Identification and security control

The extraction of morphological features for *face recognition* is proposed in Gordon and Vincent [49].

The detection of dim targets in temporal sequences of infrared images using directional *top-hat* operators is investigated in Rivest and Fortin [50]. An example of extracted targets is shown in Fig. 12.5.

Target recognition in a cluttered scene using a gray-scale generalization of the *hit-or-miss* transform is discussed in Cardillo and Sid-Ahmed [51].

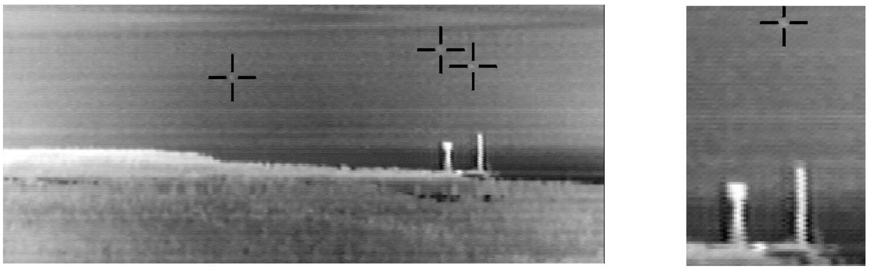


Figure 12.5: *Infrared image (left) and enlarged section (right). The targets (jets) are located at the center of the crosses. Courtesy of Dr. J.-F. Rivest, Nortel Inc., Ottawa; original image sequence by the German FIM.*

The passive detection and localization of undersea targets from sonar data using *minimal path* detection techniques is investigated in Rosen and Vincent [52].

Traffic control applications are presented in Bilodeau and Beucher [53] and Broggi [54].

There are also many papers dealing with the extraction of relevant features for the automatic recognition of *finger prints* prints, for example, Lake et al. [55].

12.7 Document processing

Due to its very nature, mathematical morphology is well suited to the extraction of morphological features of printed or manuscript characters [56, 57, 58]. A special section about the analysis of documents is available in Maragos et al. [59]. A system for recognizing musical scores is proposed in Modayur et al. [60].

The use of morphological transforms for extracting the *watermarks* of old documents is detailed in Zamperoni [61]. In particular, the author shows that top-hats are well suited to the removal of the uneven background illumination function. The *restoration* of old movies using temporal morphological filters such as closings/openings by reconstruction is introduced in Decencière Ferrandière [62, 63]. Figure 12.6 shows a damaged frame of an old movie together with the extracted defects and the restored frame.

A morphological approach for the automated segmentation of digital soil maps is proposed in Ansoult et al. [65]. The extraction of relevant features of digitized *topographic maps* using directional morphological transforms is detailed in Yamada et al. [66].

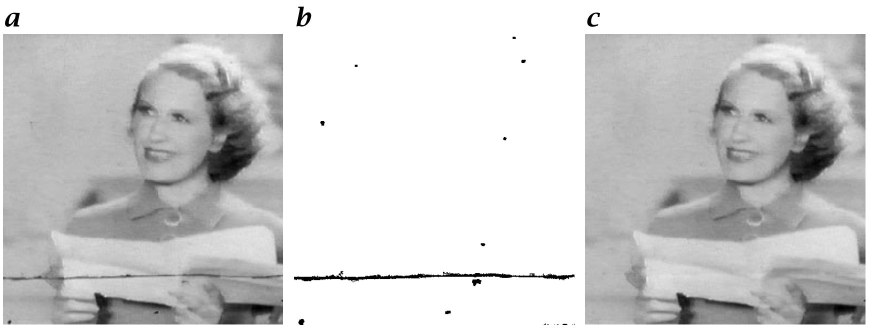


Figure 12.6: *Restoration of old motion pictures:* ***a*** *input image with crackles and blobs;* ***b*** *detected defects; and* ***c*** *restored image. By Courtesy of Dr. E. Decencière, Centre de Morphologie Mathématique of the Ecole des Mines de Paris; see also Decencière Ferrandière [64].*

12.8 Image coding

Image and video compression techniques using morphological operators such as the watershed transform, geodesic skeletons, and the interpolation technique detailed in Chapter 19 are proposed in Salembier et al. [67]. Other similar approaches are detailed in Marcotegui and Meyer [68]. An overview of region-base video coding using mathematical morphology can be found in Salembier et al. [69].

12.9 Other applications

An automatic programming system for choosing the morphological primitives by means of geometric reasoning is developed in Schmitt [70]. This author also shows in Schmitt [71] that the choice and sequencing of morphological transformations in image analysis requires expert knowledge.

An application-specific integrated circuit for morphological processing is detailed in Klein and Peyrard [72] and Peyrard [73]. For example, the block diagram of the Morphological Image Data Analysis System (MIDAS) developed at the Centre de Morphologie Mathématique (CMM) of the Ecole des Mines de Paris in the framework of a contract between the Onderzoeks Centrum voor Aanwending van Staal (OCAS) and the CMM is shown in Fig. 12.7.

In Baccar et al. [74], watersheds are used for segmenting *range images*. The segmentation function is defined as an edge image incorporating discontinuities in both surface normals and depth (i. e., roof and step edges). The marker image is obtained by thresholding the segmentation function for low gradient values.

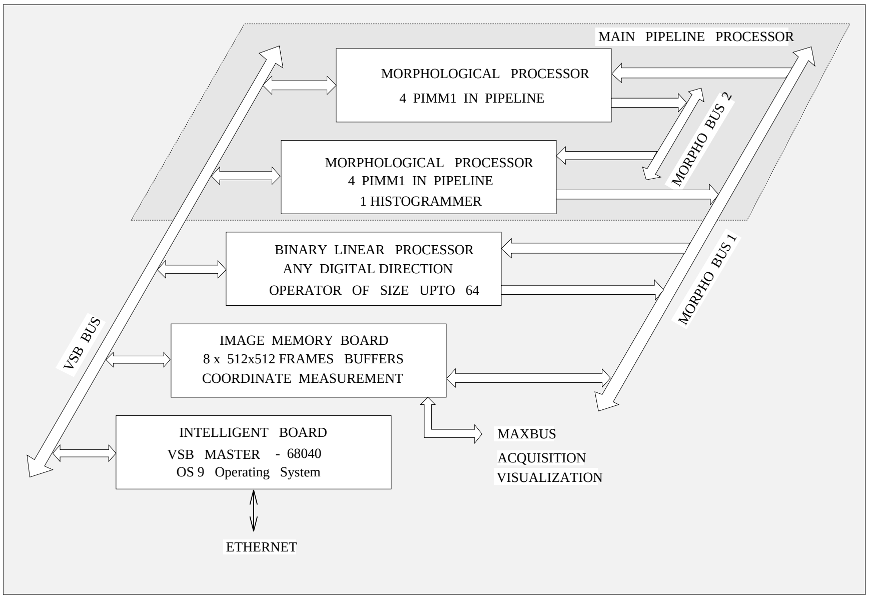


Figure 12.7: *Block diagram of the Morphological Image Data Analysis System (MIDAS), from Peyrard et al. [46].*

12.10 References

[1] Soille, P., (1998). *Morphologische Bildverarbeitung*. Berlin Heidelberg: Springer.

[2] Destival, I., (1986). Mathematical morphology applied to remote sensing. *Acta Astronautica*, **13**(6/7):371–385.

[3] Banon, G. and Barrera, J., (1989). Morphological filtering for stripping correction of SPOT images. *Photogrammetria (PRS)*, **43**:195–205.

[4] Safa, F. and Flouzat, G., (1989). Speckle removal on radar imagery based on mathematical morphology. *Signal Processing*, **16**:319–333.

[5] Martel, C., Flouzat, G., Souriau, A., and Safa, F., (1989). A morphological method of geometric analysis of images: application to the gravity anomalies in the indian ocean. *Jour. Geophysical Research*, **94**(B2):1715–1726.

[6] Lea, S. and Lybanon, M., (1993). Automated boundary delineation in infrared ocean images. *IEEE Trans. Geoscience and Remote Sensing*, **31**: 1256–1260.

[7] Watson, A., (1987). A new method of classification for LANDSAT data using the watershed algorithm. *Pattern Recognition Letters*, pp. 15–19.

[8] Watson, A., Vaughan, R., and Powell, M., (1992). Classification using the watershed method. *International Journal of Remote Sensing*, **13**(10):1881–1890.

[9] Soille, P., (1996). Morphological Partitioning of Multispectral Images. *Jour. Electronic Imaging*, **5**(3):252–265.

[10] Zheng, X., Gong, P., and Strome, M., (1995). Characterizing spatial structure of tree canopy using colour photographs and mathematical morphology. *Canadian Journal of Remote Sensing*, **21**(4):420–428.

[11] Rivest, J.-F., Beucher, S., and Delhomme, J., (1992). Marker-controlled segmentation: an application to electrical borehole imaging. *Jour. Electronic Imaging*, **1**(2):136–142.

[12] Berman, M., Bischof, L., Breen, E., and Peden, G., (1994). Image analysis for the material sciences—An overview. *Materials Forum*, **18**:1–19.

[13] Chermant, J.-L., Coster, M., and Gougeon, G., (1987). Shape analysis in R^2 using mathematical morphology. *Jour. Microscopy*, **145**(Pt 2):143–157.

[14] Coster, M. and Chermant, J.-L., (1985). *Précis d'analyse d'images*. Paris: Presses du CNRS.

[15] Jeulin, D., Vincent, L., and Serpe, G., (1992). Propagation algorithms on graphs for physical applications. *Jour. Visual Communication and Image Representation*, **3**(2):161–181.

[16] Demarty, C.-H., Grillon, F., and Jeulin, D., (1996). Study of the contact permeability between rough surfaces from confocal microscopy. *Microsc. Microanal. Microstruct.*, **7**:505–511.

[17] Talbot, H., (1996). A morphological algorithm for linear segment detection. In *Mathematical morphology and its applications to image and signal processing*, P. Maragos, R. Schafer, and M. Butt, eds., pp. 219–226. Boston: Kluwer Academic Publishers.

[18] Talbot, H., Jeulin, D., and Hanton, D., (1996). Image analysis of insulation mineral fibers. *Microsc. Microanal. Microstruct.*, **7**:361–368.

[19] Talbot, H., (1993). *Analyse morphologique de fibres minérales d'isolation*. PhD thesis, Ecole des Mines de Paris.

[20] Viero, T. and Jeulin, D., (1995). Morphological extraction of line networks from noisy low-contrast images. *Jour. Visual Communication and Image Representation*, **6**(4):335–347.

[21] Pirard, E., (1994). Shape processing and analysis using the calypter. *Jour. Microscopy*, **175**(3):214–221.

[22] Jeulin, D., Terol Villalobos, I., and Dubus, A., (1995). Morphological analysis of UO_2 powder using a dead leaves models. *Microscopy, Microanalysis, Microstructure*, **6**:371–384.

[23] Gratin, C. and Meyer, F., (1992). Mathematical morphology in three dimensions. In *8th Int. Congress for Stereology*, pp. 551–558.

[24] Meyer, F., (1979). Iterative image transformations for an automatic screening of cervical smears. *Jour. Histochemistry and Cytochemistry*, **27**:128–135.

[25] Skolnick, M., (1986). Application of morphological transformations to the analysis of two-dimensional electrophoretic gels of biological materials. *Computer Vision, Graphics, and Image Processing*, **35**:306–332.

[26] Gratin, C., (1993). *De la représentation des images au traitement morphologique d'images tridimensionnelles.* PhD thesis, Ecole des Mines de Paris.

[27] Conan, V., Gesbert, S., Howard, C., Jeulin, D., and Meyer, F., (1991). Geostatistical and morphological methods applied to three-dimensional microscopy. *Jour. Microscopy*, **166-2**:169–184.

[28] Higgins, W. and Ojard, E., (1993). Interactive morphological watershed analysis for 3D medical images. *Computerized Medical Imaging and Graphics*, **17**(4/5):387–395.

[29] Zehetbauer, S. and Meyer-Gruhl, U., (1993). Segmentierung und Analyse drei- und vierdimensionaler Ultraschalldatensätze. In *Mustererkennung 1993*, S. Pöppl and H. Handels, eds., pp. 119–125. Springer-Verlag.

[30] Matsopoulos, G., Marshall, S., and Brunt, J., (1994). Multiresolution morphological fusion of MR and CT images of the human brain. *IEE Proc.-Vis. Image Signal Process.*, **141**:3137–142.

[31] Matsopoulos, G. and Marshall, S., (1995). Application of morphological pyramids: fusion of MR and CT phantoms. *Jour. Visual Communication and Image Representation*, **6**(2):196–207.

[32] Casas, J., Esteban, P., Moreno, A., and Carrera, M., (1994). Morphological scheme for morphometric analysis of epidermal biopsy images. In *Mathematical morphology and its applications to image processing*, J. Serra and P. Soille, eds., pp. 325–331. Dordrecht: Kluwer Academic Publishers.

[33] Vachier, C., (1995). *Extraction de caractéristiques, segmentation d'images et morphologie mathématique.* PhD thesis, Ecole des Mines de Paris.

[34] Vincent, L. and Masters, B., (1992). Morphological image processing and network analysis of cornea endothelial cell images. In *Image algebra and morphological image processing III*, P. Gader, E. Dougherty, and J. Serra, eds., Vol. SPIE-1769, pp. 212–226.

[35] Pina, P., Selmaoui, N., and Amaral Fortes, M., (1996). Geometrical and topological characterization of cork cells by digital image analysis. In *Mathematical morphology and its applications to image and signal processing*, P. Maragos, R. Schafer, and M. Butt, eds., pp. 459–466. Boston: Kluwer Academic Publishers.

[36] Thiran, J.-P. and Macq, B., (1996). Morphological feature extraction for the classification of digital images of cancerous tissues. *IEEE Trans. Biomedical Engineering*, **43**(10):1011–1020.

[37] Manders, E., Hoebe, R., Stackee, J., Vossepoel, A., and Aten, J., (1996). Largest contour segmentation: a tool for the localization of spots in confocal images. *Cytometry*, **23**:15–21.

[38] Laitinen, T., Silven, O., and Pietikäinen, M., (1990). Morphological image processing for automated metal strip inspection. In *Image Algebra and Morphological Image Processing*, P. Gader, ed., Vol. SPIE-1350, pp. 241–250.

[39] Jochems, T., (1994). *Morphologie mathématique appliquée au contrôle industriel de pièces coulées.* PhD thesis, Ecole des Mines de Paris.

[40] Jochems, T. and Préjean-Lefèvre, V., (1993). Mathematische Morphologie in der Praxis: Konstruktion eines Algorithmus für die Erkennung von Produktionsfehlern in Turbinenschaufeln. *Vision & Voice Magazine*, **7**(1): 8–15.

[41] Müller, S. and Nickolay, B., (1994). Morphological image processing for the recognition of surface defects. In *Automated 3-D and 2-D Vision*, R.-J. Ahlers, D. Braggins, and G. Kamerman, eds., Vol. SPIE-2249, pp. 298–307.

[42] Whelan, P. and Batchelor, B., (1993). Flexible packing of arbitrary two-dimensional shapes. *Optical Engineering*, **32**(12):3278–3287.

[43] Whelan, P. and Batchelor, B., (1996). Automated packing systems: A systems engineering approach. *IEEE Trans. Systems, Man and Cybernetics*, **26**(5):533–544.

[44] Whelan, P. and Soille, P., (Submitted, 1998). Watermark extraction in paper samples. In *Irish Machine Vision and Image Processing (IMVIP'98)*, D. Vernon, ed., pp. 287–299. Maynooth.

[45] Tuzikov, A., Soille, P., Jeulin, D., Bruneel, H., and Vermeulen, M., (1992). Extraction of grid lines on stamped metal pieces using mathematical morphology. In *Proc. 11th IAPR International Conference on Pattern Recognition, Conference A: Computer Vision and Applications*, Vol. 1, pp. 425–428. The Hague.

[46] Peyrard, R., Soille, P., Klein, J.-C., and Tuzikov, A., (1995). A dedicated hardware system for the extraction of grid patterns on stamped metal sheets. In *Proc. of 1995 IEEE Workshop on Nonlinear Signal and Image Processing*, I. Pitas, ed., pp. 867–870. Neos Marmaras. URL: http://poseidon.csd.auth.gr/Workshop/papers/p_34_3.html.

[47] Soille, P., (1998). Morphological unwrapping of interferometric phase-map. In *Mathematical morphology and its applications to signal and image processing IV*, H. Heijmans and J. Roerdink, eds., pp. 383–390. Dordrecht: Kluwer Academic Publishers.

[48] Breen, E., Jones, R., and Talbot, H., (1998). The morphological approach to industrial image analysis applications. *Acta Stereologica*.

[49] Gordon, G. and Vincent, L., (1992). Application of morphology to feature extraction for face recognition. In *Nonlinear Image Processing III*, E. Dougherty, J. Astola, and G. Boncelet, eds., Vol. SPIE-1658, pp. 151–164.

[50] Rivest, J.-F. and Fortin, R., (1996). Detection of dim targets in digital infrared imagery by morphological image processing. *Optical Engineering*, **35**(7):1886–1893.

[51] Cardillo, J. and Sid-Ahmed, M., (1996). Target recognition in a cluttered scene using mathematical morphology. *Pattern Recognition*, **29**(1):27–49.

[52] Rosen, B. and Vincent, L., (1994). Morphological Image Processing Techniques Applied to Detection of Correlogram Tracks. *U.S. Navy Journal of Underwater Acoustics*, **44**(2):571–586.

[53] Bilodeau, M. and Beucher, S., (1994). Road segmentation using a fast watershed algorithm. In *ISMM'94: Mathematical morphology and its appli-*

cations to image processing —Poster Contributions—, J. Serra and P. Soille, eds., pp. 29–30. Ecole des Mines de Paris.

[54] Broggi, A., (1995). Parallel and local feature extraction: a real-time approach to road boundary detection. *IEEE Trans. Image Processing*, **4**(2): 217–223.

[55] Lake, C., Lougheed, R., and Beyer, J., (1993). Morphological algorithm for ridge extraction in fingerprint images. In *Image algebra and morphological image processing IV*, E. Dougherty, P. Gader, and J. Serra, eds., Vol. SPIE-2030, pp. 334–345.

[56] Bloomberg, D. and Maragos, P., (1990). Generalized hit-miss operations. In *Image Algebra and Morphological Image Processing*, P. Gader, ed., Vol. SPIE-1350, pp. 116–128.

[57] Bloomberg, D. and Vincent, L., (1995). Blur hit-miss transform and its use in document image pattern detection. In *Document Recognition II*, L. Vincent and H. Baird, eds., Vol. SPIE-2422, pp. 278–292.

[58] Liang, S., Ahmadi, M., and Shridhar, M., (1994). A morphological approach to text string extraction from regular periodic overlapping text/background images. *Computer Vision, Graphics and Image Processing: Graphical Models and Image Processing*, **56**(5):402–413.

[59] Maragos, P., Schaffer, W., and Butt, M. (eds.), (1996). *Mathematical Morphology and its Applications to Image and Signal Processing*. Kluwer Academic Publishers.

[60] Modayur, B., Ramesh, V., Haralick, R., and Shapiro, L., (1993). MUSER: a prototype musical score recognition system using mathematical morphology. *Machine Vision and Applications*, **6**:140–150.

[61] Zamperoni, P., (1989). Wasserzeichenextraktion aus digitalisierten Bildern mit Methoden der digitalen Bildsignalverarbeitung. *Das Papier*, **43**(4):133–143.

[62] Decencière Ferrandière, E., (1996). Motion picture restoration using morphological tools. In *Mathematical morphology and its applications to image and signal processing*, P. Maragos, R. Schafer, and M. Butt, eds., pp. 361–368. Boston: Kluwer Academic Publishers.

[63] Decencière Ferrandière, E., (1996). Restoration of old motion pictures. *Microscopy, Microanalysis, Microstructures*, **7(5/6)**:311–316.

[64] Decencière Ferrandière, E., (1997). *Restauration automatique de films anciens*. PhD thesis, Ecole des Mines de Paris.

[65] Ansoult, M., Soille, P., and Loodts, J., (1990). Mathematical morphology: a tool for automated GIS data acquisition from scanned thematic maps. *Photogrammetric Engineering and Remote Sensing*, **56**(9):1263–1271.

[66] Yamada, H., Yamamoto, K., and Hosokawa, K., (1993). Directional mathematical morphology and reformalized Hough transformation for the analysis of topographic maps. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **15**(4):380–387.

[67] Salembier, P., Brigger, P., Casas, J., and Pardàs, M., (1996). Morphological operators for image and video compression. *IEEE Trans. Image Processing*, **5**(6):881–898.

[68] Marcotegui, B. and Meyer, F., (1994). Morphological segmentation of image sequences. In *Mathematical morphology and its applications to image processing*, J. Serra and P. Soille, eds., pp. 101–108. Dordrecht: Kluwer Academic Publishers.

[69] Salembier, P., Torres, L., Meyer, F., and Gu, C., (1995). Region-based video coding using mathematical morphology. *Proceedings of IEEE*, **83**(6):843–857.

[70] Schmitt, M., (1989). Mathematical morphology and artificial intelligence: an automatic programming system. *Signal Processing*, **16**:389–401.

[71] Schmitt, M., (1991). Variations on a theme in binary morphology. *Jour. Visual Communication and Image Representation*, **2**(3):244–258.

[72] Klein, J.-C. and Peyrard, R., (1989). PIMM1, an Image Processing ASIC Based on Mathematical Morphology. In *IEEE's ASIC Seminar and Exhibit*, pp. 25–28. Rochester NY.

[73] Peyrard, R., (1992). *Conception et mise en oeuvre d'un A.S.I.C de morphologie mathématique à architecture programmable.* PhD thesis, Ecole des Mines de Paris.

[74] Baccar, M., Gee, L., Gonzalez, R., and Abidi, M., (1996). Segmentation of range images via data fusion and morphological watersheds. *Pattern Recognition*, **29**(10):1673–1687.

13 Industrial Object Recognition

Thomas Wagner and Peter Plankensteiner

Fraunhofer Institut für Integrierte Schaltungen, Erlangen, Germany

13.1 Problem, market and solutions

13.1.1 Introduction

In this application note, we will demonstrate an object recognition approach for industrial recognition tasks. We focus on three performance features: manageability, many object classes, and speed.

At first, we present a teachable optical quality control system called Intelli-Cam, which is based on a so-called intelligent camera hardware, and give examples of different application setups.

Furthermore, we briefly summarize current hard- and software developments in *intelligent cameras*, which will, before long, lead to realtime evaluation of some 100 frames/s.

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

As an outlook, we extend object recognition to industrial environments with very many (i. e., hundreds, thousands, or even more) different object classes. We provide benchmark data sets for such object recognition tasks, and give recognition results for different types of object features used in evaluation.

13.1.2 Problem description

Object recognition is one of the basic problems in industrial image analysis. Though in principle the task was solved years ago, from a practical point of view a number of difficulties have not yet been eliminated. By examining the effect of those difficulties in detail, a deep insight into the intrinsic problems of a given recognition setup can be gained in advance.

First of all, the *number of objects* in a single image is of crucial importance. Setups for automatic identification in industrial environments try to separate single objects prior to image acquisition, for example, by means of mechanics. Images with more than one object are always more demanding.

The most prominent problem in automatic identification is, as for many other image processing tasks, the *imbalance in illumination* in time or in space. Variations in time stem from changing daylight conditions and aging effects of or defects in the artificial illumination system. Global illumination changes in space are primarily caused by either insufficient balance of the artificial illumination system (it is not at all an easy task to distribute enough light evenly to an area of half a meter in diameter). Local variations may be caused by shadows or reflections (which depend very much on the underlying type of surface material). Solutions to these problems are discussed in Volume 1, Chapter 6.

Another aspect of the problem's demand is the task of *object-background separation*. Obviously, it is much more difficult to separate an object from a varying background than from a constant one. In this context, it is not important whether the variations take place in time or in space. In the worst case, both variations occur simultaneously. In the best case, a locally varying background, which is constant in time, can be subtracted from the image.

More hindering effects come from the variable position and orientation of the object in space relative to the camera system. This causes *variability in position, aspect, or scale*. The problem becomes even more difficult if the object itself is not rigid, such that deformations become possible, or if different representatives of an object class are per se not identical. It will be easy to detect, for example, a special type of pencil automatically. In can already be more demanding to reliably identify an apple within other fruits. With the current state of the art, it is impossible to automatically decide whether an unknown object belongs

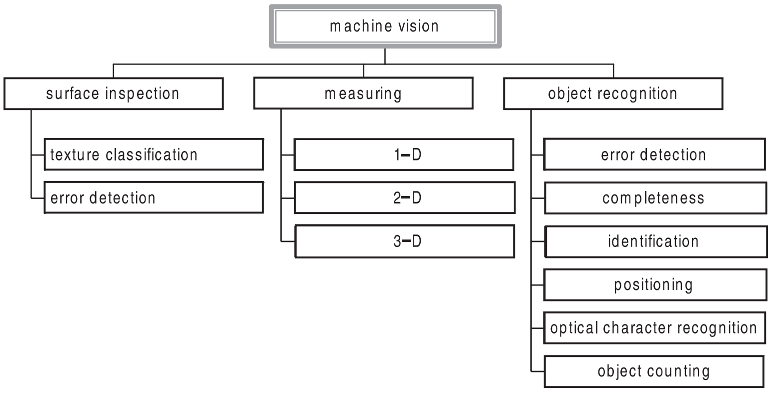


***Figure 13.1:** Object recognition is an important part of industrial image processing.*

to a quite general object class such as "chair." Methods for generalizations on a high level of abstraction which could cope with this type of problem are not yet known and neither is the way the human brain tackles this problem. *Hiding effects* also raise problems in automatic image analysis, whether caused by other objects or dirt. The *recognition speed* is critical in cases where production proceeds with rates of more than 25 parts/s, as the video frame rate of standard CCD cameras limits image acquisition speed. The *total number of object classes* to be recognized affects the difficulty of the task. This aspect will be discussed in Section 13.3.

A professional image processing system has to meet a number of requirements resulting from special industrial environment conditions. The four most important criteria are: Insensitivity with respect to disturbances (dirt and dust, EMV-pulses, change in illumination, etc.), speed, price (initial price and "cost of ownership"), and variability. Depending on the application, there will be different priorities within these criteria.

Furthermore, it is obvious that the hardware of a system must be protected against the industrial environment. Short periods for training and classification as well as a simple and robust handling are requested. This is especially true for teaching new objects to the system.

13.1.3 Market analysis and solutions

Object recognition is an important application field in industrial image processing (Fig. 13.1). In a market with an estimated annual turnover of 550 million U.S. dollars in 1998 for Europe and an annual growth rate of 20 %, about 26,000 systems are sold per year ([1], see also Chapter 11).

In a study on the question of application fields for image processing systems [1], the most frequent replies were the fields “examination of contour,” “positioning,” “measuring,” and “object identification.” For all of these tasks, object recognition methods are brought into action (see Section 11.4.3).

Solutions for object recognition tasks are either PC-based, or they consist of special hardware. For a long time, special hardware was equivalent to the use of boards in VME-bus or PC-bus technology. Meanwhile, for the field of object recognition, a third alternative has been developed, so-called intelligent cameras. The idea behind intelligent cameras is to develop a stand-alone product containing camera and processor. Intelligent cameras can communicate directly with the production line by the use of dedicated interfaces. Such cameras are described in more detail in the following section.

13.2 Compact solution: intelligent cameras

13.2.1 Overview

An intelligent camera integrates a sensor and a processing unit within the camera chassis and, therefore, requires a minimum of space. Furthermore, due to the direct processing in the camera, potential bottlenecks such as the PC bus are avoided.

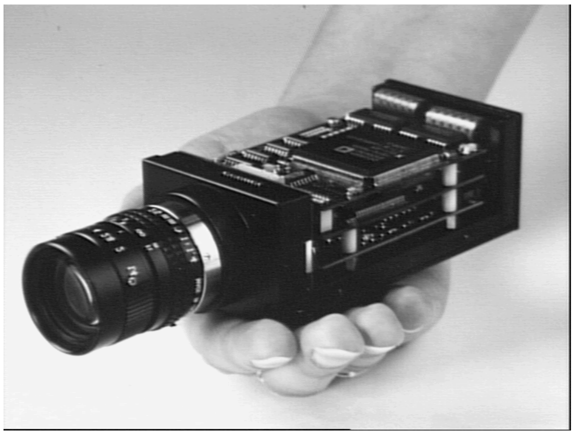
The state of the art knows three basic construction principles for intelligent cameras:

- The integration of PC boards in a camera is relatively simple. The development can be done with the usual compilers and tools. However, the resulting system requires some space, and hardware costs are relatively high.
- The use of a special DSP processor tends to increase speed and decrease the volume of an intelligent camera system.
- Integration of sensor and processor on a single chip (e. g., in CMOS technology) can further boost processing speed and decrease system size. However, a drawback of such technology is that such systems are not as flexible as programmable processors.

More information on some examples of intelligent cameras can be found in the web. The VE262 smart camera from VEtech (`http://www.vetech.com`) incorporates a monochrome camera with a 486 PC. Another example of integrated PC is the PCCAM from maVis(`http://www.ziam.de/NRW-IPD/mavis/dwelcome.ht`).

Intelligent cameras combined with a special hardware architecture and processor can be found by the imputer 3 from Vision (`www.vvl.co.uk/imputer`), smartEYE from Phytec (`www.phytec.de`), NANOcam from

a



b

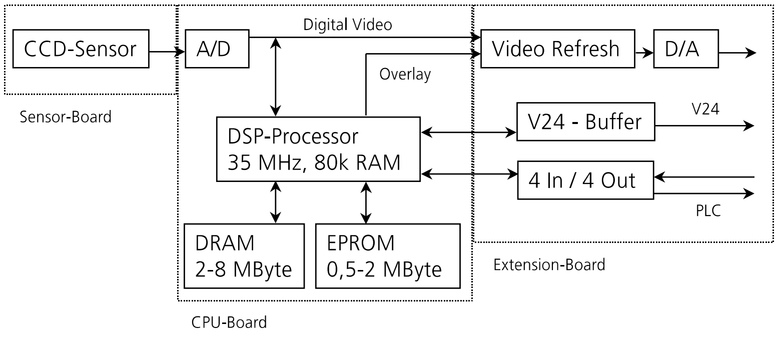


Figure 13.2: ***a*** *The VC21 camera: a complete image processing system within a volume of 180 cm*^{3}*;* ***b*** *hardware setup of the Intelli-Cam system.*

NANOsystems (www.nanosystems.de), and the VC-series from Vision Components (www.vision-components.de).

The SmartVision Sensor from IVP (www.ivp.se) is a combination of sensor and general purpose image processor on the same CMOS chip. Although the technology exists to perform image preprocessing directly in the sensor, no industrial applications have been realized to date.

13.2.2 An example: the intelligent camera system Intelli-Cam

To satisfy the conditions of a robust image processing system, we conceived the teachable system Intelli-Cam, which works as a stand-alone system on the basis of the intelligent DSP-Camera VC21(Vision Components GmbH, http://www.vision.components.de, Fig. 13.2a).

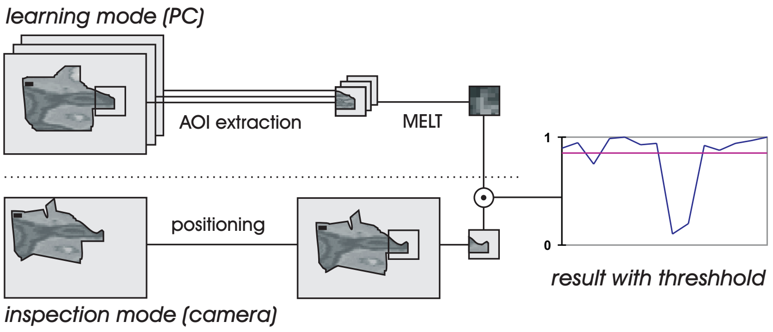
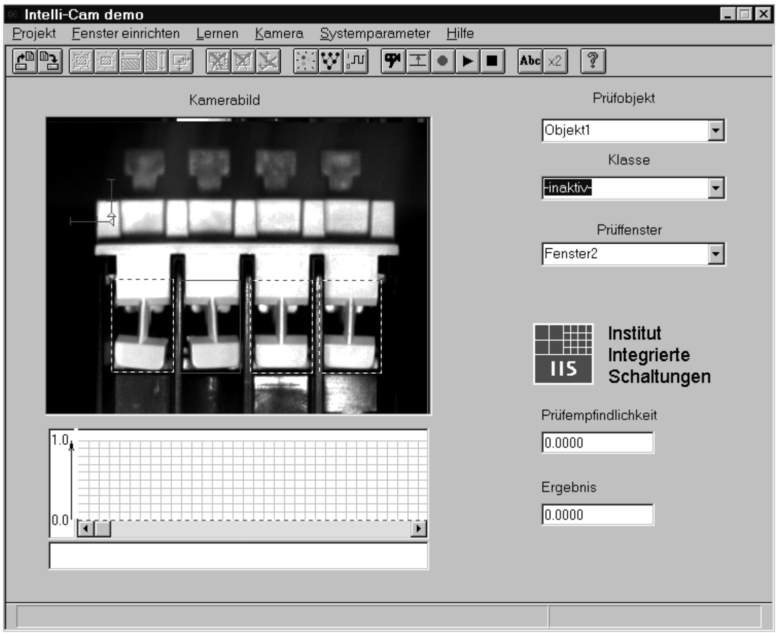


Figure 13.3: *The training and the inspection process in Intelli-Cam.*

The hardware of the VC21 camera (Fig. 13.2b) consists of a CCD sensor with a resolution of 752×582 pixels in 8 bit. An ADSP2181 processor from Analog Devices, 2 to 8 MByte DRAM, and 0.5 to 2 MByte EPROM allow direct evaluation of the images as well as permanent storage of the evaluation programs. Communication for external parameterization (e. g., by means of a portable PC or laptop) is established via a serial interface, while 4 binary input/output channels allow direct contact with the process control in the production line. Additionally, a video output for monitoring purposes is provided.

In connection with a special evaluation software, the camera is commercially available as the Intelli-Cam system (CamControl GmbH, `http://home.t-online.de/home/cam-control-nbg`). The idea of Intelli-Cam is to provide a flexible low-cost system for simple industrial image processing tasks. No complex set-up procedure should be necessary to adapt Intelli-Cam to a special image processing problem. Therefore, we avoided the long-winded adjustment of a large number of parameters, which is a disadvantage of many commercial state-of-the-art image processing systems. With just *one* single sensitivity parameter for each inspection problem, even laymen can carry out the installation of Intelli-Cam quickly.

To adapt the Intelli-Cam System to a special image processing problem, only a few steps have to be performed (Fig. 13.3). The training is done by a PC-based application providing a graphical user interface (Fig. 13.4) for a straightforward parametrization of the system. After marking the relevant regions in an image, the internal training via the synergetic MELT algorithm [2] is started. This algorithm has shown to be very robust in industrial applications. High recognition rates, short classification times, and especially short training periods are its most prominent features.



Prüfobjekt: Objekt1
Klasse: -inaktiv-
Prüffenster: Fenster2
Prüfempfindlichkeit: 0.0000
Ergebnis: 0.0000

Figure 13.4: *Graphical user interface for the parameterization of Intelli-Cam.*

Finally, the user has to define a rejection threshold. From that moment, the camera works in a stand-alone mode, and the PC may be disconnected.

A significant preprocessing step in pattern recognition is the detection of the precise position of the inspected object within the image. Even a small displacement or rotation of the object can cause unpredictable results in the recognition process. To submit a fast and precise tracking of the tested object within the image, we devised a two-step strategy:

- Coarse positioning of the inspected object by the help of up to two templates.
- Fine positioning of the inspected regions in a small neighborhood.

Both methods together allow precise positioning, but are very time extensive. To overcome this restriction, we reduce the image resolution by generating the first level of an image pyramid. With subpixel interpolation methods [3], an exact position can be achieved, while a speedup of nearly a factor of 16 is realized.

Intelli-Cam targets at a variety of comparably easy inspection tasks. The most important applications are good/bad decisions as well as sorting problems. The good/bad classification mode allows us to reject de-

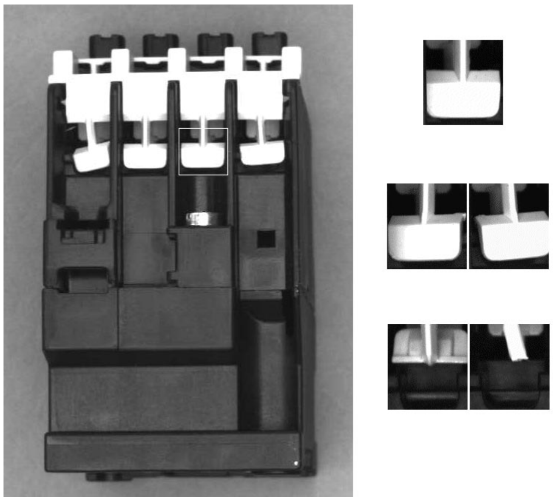


Figure 13.5: *A contactor as an example of a typical inspection problem and the relevant regions in detail: One correct plate, two typical minor faults, and two major defects.*

fect parts in an industrial production process, for example, sorting out circuit boards with missing chips. In this case, the user trains the system with just one object class representing an example of the faultless object. The sensitivity threshold defines the maximal acceptable difference to the test object. Defining a weaker threshold allows greater variations of the tested object.

An application for Intelli-Cam is given in Fig. 13.5. Contacts on relays must be correctly covered in order to protect users from electric shocks. At the end of the production process, some of these plastic plates are distorted or completely damaged. To avoid risks for users, the defect contactors must be detected and removed. The figure gives an example of a correct plate and some typical defects.

The total time for the examination of one contactor is about 320 ms, including time for frame grabbing and positioning. Classification alone takes about 60 ms. The training procedure took about 90 s on a Pentium with 90 MHz. The choice of the sensitivity threshold allows for the decision as to whether parts with slight defects are rejected by the Intelli-Cam system, or whether they pass the examination.

In applications with more than one object class, object identification is performed. Such an example is shown in Fig. 13.6. Different types of empties on a band-conveyor have to be identified automatically. A special difficulty of the problem is the fact that the boxes may not be completely filled, that is, boxes of the same class may contain varying numbers of bottles.

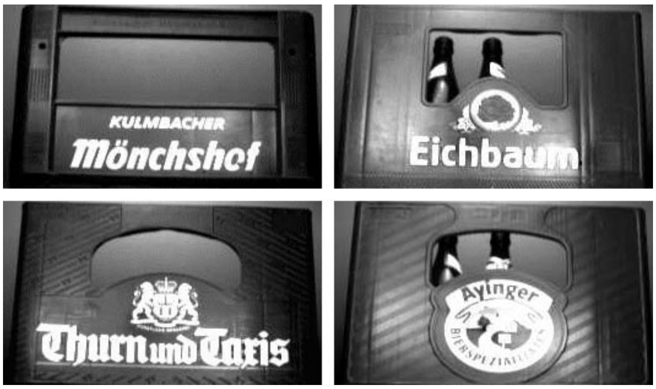


Figure 13.6: *Different types of empties.*

13.2.3 FastCam: high-speed object identification

Real-time image processing systems use to work at frame rates of 25 full frames/s. Digital video systems with higher performance are limited to an image acquisition mode, followed by visual playback and an off-line evaluation. An example of such a system is the Speed-Cam system, which allows 1000 images/s with a resolution of 512×512 gray-level pixel (see: `http://www.weinberger.ch/speedcam.htm`).

In order to bridge that gap, we have developed the FastCam system, which allows an online inspection with frame rates of up to 470 frames/s.

The camera hardware works with a resolution of 512 x 512 pixel at 24 MHz. The readout of full, half, quarter, and eighth of a frame generates frame rates of 115 to 670 Hz. The processing unit consists of a PC Dual Pentium II with 300 MHz.

Simple algorithms have to be used to allow an online high-speed inspection. As a consequence, operations such as filters, thresholds, border extractions etc. are brought into action.

13.3 Object recognition for many object types

13.3.1 The problem

In many applications of industrial object recognition, the number of object types that have to be distinguished is quite limited. A typical number of classes lies between 10 and 100. However, there are tasks in which this limit is drastically exceeded. Some examples shall be given.

Object recognition in supermarkets is one of them. A typical market offers between 10,000 and 100,000 different articles. Although they are usually identified by their bar code, there are similar setups where the use of such codes may not be possible. One could, for example, think of attempts to automate the function of the cashier. A prerequisite for this is that there is no longer a need for handling the products in order to find the bar code's position. One way to achieve that would be to find the bar code automatically by means of image processing, the other one would be to identify the object by other typical features, without the need of locating the bar code.

Another field of application with a large number of object classes are stores with goods that do not possess a bar code. An example would be stores for replacement parts in the automotive industry, especially for subcontractors. Those people are usually shown a part to be replaced, and have to locate the adequate replacement part in their stock.

A third example stems from the print industry. Books, papers, and journals are typically sold on a commission basis, and the publishers refund money for products that have not been sold. As a consequence, there is a need for automatically sorting the returned goods, in order to settle accounts with the distributors. In this case, an additional technical problem stems from the fact that the returned articles may be dirty or modified, for example, by labels such as "special bargain" etc.

If one attempts to tackle such problems with conventional methods of object recognition, one will find that the recognition rates are usually not sufficient. To put it the other way round, for a given method, recognition rates decrease with an increasing number of object types to be identified. The reason for this effect is obvious: while the limited number of object features typically used in conventional object recognition setups may be sufficient to tackle applications with a restricted number of object types, they no longer provide enough relevant information when the number of objects is increasing.

The solution is also straightforward: increase the number and the type of object features, that is, use color, shape, texture etc.

A disadvantage is the known fact that the performance of a system decreases with an increasing number of *irrelevant* object features included. As a consequence, one has to implement methods of determining *relevant* object features for a given problem while increasing the number of object features used.

13.3.2 Data sets

Two data samples have been used for the subsequent experiments. Table 13.1 gives an overview of their specifications.

The first sample stems from objects in a supermarket. Images of 1000 different object types have been acquired. Per object type, three

Table 13.1: *Datasets for benchmarking*

		Supermarket	Replacement parts
Abbreviation		SUP_1000	AUT_331
classes		1000	331
Patterns per class			
	Training	1	1
	Testing	1	1
	Verification	1	2
Image size		768×576	512×512
Bits per pixel		3×8	3×8 (see footnote [1])

different representatives were selected and one image of each one was grabbed. All objects were positioned with their main axis of inertia in an approximately horizontal direction. The sample was divided in a training, a testing, and a verification sample.

The second sample set stems from a stock of replacement parts for cars. While the stock contains some 10,000 objects, images from a selection of 331 have been acquired. Object examples from both sample sets are shown in Fig. 13.7.

13.3.3 Image features for object recognition

The setup for the object recognition system with which we solved the problem is shown in Fig. 13.8. It is a feature-based recognition approach.

In a first step, the objects in the image are separated from the background. For this task either threshold-based methods or more advanced segmentation techniques are used.

In a second step, a number of different feature sets are extracted from the segmented images: In order to obtain textural features, which have been described in detail in Volume 2, Chapter 12, the image is rotated until the major axis of the object is aligned horizontally. The textural features are then determined in the bounding box enclosing the object. In addition, the color and geometric features listed in Table 13.2 are determined from the segmented images. Chapter 12 in

[1]Due to camera restrictions, the *de facto* color resolution of the color images is about 3×5 bit per pixel.

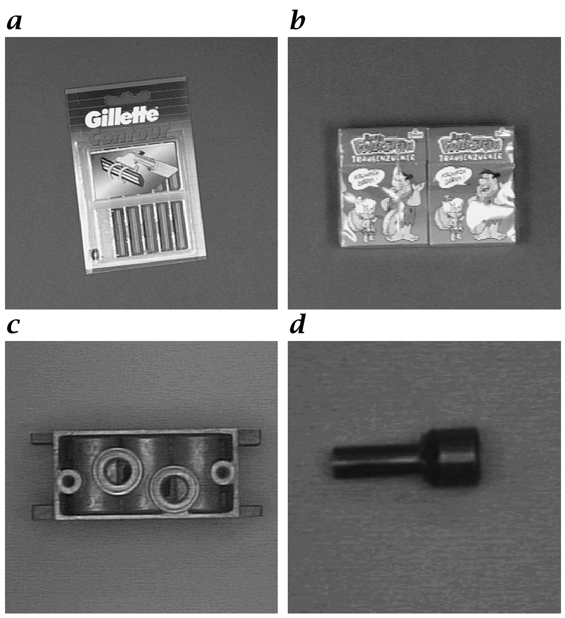


Figure 13.7: *a and b Objects from the supermarket sample; c and d replacement parts from the car supplier sample.*

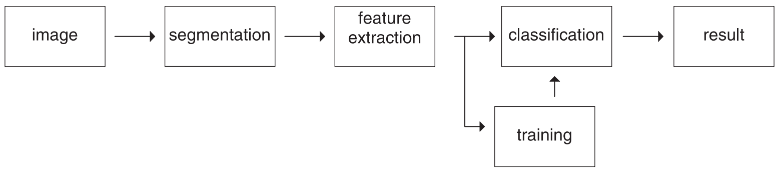


Figure 13.8: *Typical setup of an object recognition experiment: features are extracted from a segmented image and put into a classifier.*

Volume 2 lists the detailed calculation times[2] necessary to extract textural features.

The features calculated from the training sample are used to train a classifier, in our case a 1-nearest neighbor classifier. The recognition rate is determined on the verification sample set, while an additional testing sample set serves as a means of determining intermediate recognition rates when performing automatic feature selection.

As described in Section 13.3.1, it is essential to use the most adequate features for a given data set in order to achieve high recognition accuracy. As a consequence, an established feature selection technique

[2]Computation times have been measured on a Pentium 200 MHz PC without MMX under Linux, Kernel Version 2.0.22 using the GNU g++ compiler, version 2.7.2.1. The PC achieved a linpack benchmark of 11,4 MFLOPS with the java applet [4].

has been applied to select subsets of optimal features for each of the two given data sets.

The feature selection technique works as follows. For a given training and testing set, calculate all feature values on all images. In a first selection step, select the single feature with the highest recognition rate. Here and in the following, the recognition rate is determined on the testing sample, while the training sample serves to teach the classifier in use.

In a next step, determine the recognition rates for all feature subsets with two features that contain the feature of step one. Do the same with a third feature.

Instead of directly adding a fourth feature, remove that one of the three selected features with the least significant influence on the recognition rate (i. e., for which the recognition rate decreases minimally after the feature has been removed).

Continue adding further features in a similar manner, always trying to increase the recognition rate in maximal steps when adding an additional feature, and always removing one of all features after having added three.

Determine the feature set with the highest recognition rate. Alternatively, the (quite time consuming) selection process can be stopped when there is no longer a significant increase in recognition rates.

For an independent check of the robustness of the selected feature subset, determine the recognition rate on a third data sample, the verification sample. Obviously, this recognition rate will be lower, as the selection is optimal with respect to the testing sample. Nevertheless, for sample sets of sufficient size, generalization has taken place, and the performance is still significantly better than on unselected feature sets.

We just mention that there are other selection techniques, which are based on methods such as measures of correlation for different features or genetic algorithms.

13.3.4 Performance and results

An overview on the performance of the different feature sets is given in Table 13.3. Recognition rates vary considerably between different types of feature sets, as do the computing times for the features.

When identifying objects, one usually wants to recognize the object class exactly. For a great number of objects, however, it may be sufficient to find the object in the K classes with the best match to the testing sample, and to do the exact identification manually. Therefore, recognition rates for different parameter values of K are also given in the table.

Table 13.2: *Features for object recognition: In addition to the textural features described in detail in Volume 2, Chapter 12, color and geometric features are used for robust identification*

Feature set	No.	Principle
Textural features	318	texture analysis (Volume 2, Chapter 12)
Average RGB values	3	color analysis
Gradient method	32	mean of the Sobel image in 32 orientations
Symmetry method	32	measure for symmetry in 32 orientations
Central moments	7	central moments up to third order
Object area A	1	area without holes
Width and height	2	
Perimeter p	1	
Waddel disk diameter R_d	1	$R_d = 2\sqrt{A}/\sqrt{\pi}$
Hydraulic radius R_h	1	$R_h = A/p$
Heywood circularity factor F_h	1	$F_h = p/(2\sqrt{\pi A})$
Ellipse axes E_{2a}, E_{2b}	2	minor and major axis of ellipse with same A and p
Ellipse ratio E_{ab}	1	$E_{ab} = E_{2a}/E_{2b}$
Rectangle width and height R_c, r_c	2	big and small side of the rectangle with same A and p
Rectangle ratio Rr_c	1	$Rr_c = R_c/r_c$
Compactness F_c	1	$F_c = A/(\text{width} \times \text{height})$ (width and height: max. object extension)
Number of holes	1	
Area of holes	1	
Chords X and Y	2	maximal number of chords in x and y
Maximum chords	2	maximal length of chords in x and y
Maximum intercept	1	

It is remarkable that the performance of many of the textural features beats the shape-based features. The color features, which are based on a simple computation, also perform well. The feature subsets at the end of Table 13.3 confirm this observation: The shape features are less expensive in terms of computation time, but at the same time they do not yield sufficiently high recognition rates.

It can also be seen from the results that the recognition on the basis of all features implemented does not boost the recognition rate at all. The performance stagnates when adding new features; quite often a

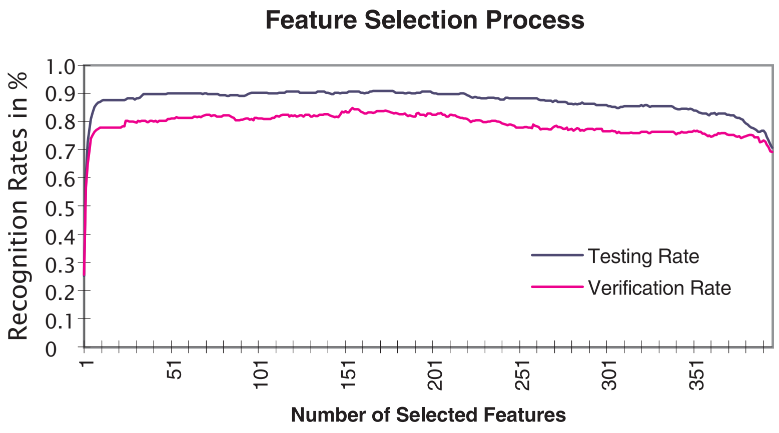


Figure 13.9: *The process of automatic feature selection: increasing recognition rates during the beginning of the selection process lead to a performance maximum. For large numbers of selected features, the performance decreases again.*

reduction is even observed. It is a well-known effect that irrelevant features reduce recognition rates.

We wanted to find out to what extent a feature selection process can improve the recognition results further. Figure 13.9 shows the recognition performance of a selection process on the AUT_331 sample. Values for both testing and verification sample are shown. The results for the testing sample are obviously higher, as that sample has been used for optimization. The performance graphs show a maximum that is reached quite early, and decrease again for large feature subsets.

As a result, Fig. 13.10 summarizes the performances for different approaches. The combination of the different feature types—texture, color, and shape alone—does not increase recognition rate. However, making use of a feature selection technique helps to improve recognition considerably (in our experiments, approximately 10 %).

13.4 Discussion and outlook

With the Intelli-Cam system, we have shown a successful example for an off-the-shelf image processing product for simple object recognition tasks. The success of such products lies in their standardization, which in consequence leads to a competitive relation of price and features.

The FastCam system, aiming at inspection problems with frame rates of a few hundred images per second, proves that there is still room for hardware development in image processing systems, as long as the solutions generated cannot be beaten with conventional PC-based

***Table 13.3:** Benchmark results: feature extraction time and recognition rates for the supermarket (1000 classes) and replacement parts (331 classes) data sets*

	Time in s	supermarket			replacement parts		
K =		1	5	10	1	5	10
Texture							
Haralick	*0.56*	74.1	89.4	93.7	52.2	73.4	79.9
Unser	*0.58*	78.3	92.0	95.3	45.5	70.4	77.9
Galloway	*0.33*	60.0	80.5	86.2	34.3	60.7	70.8
Laine	*2.77*	74.9	89.6	93.0	34.7	57.7	67.5
Local	*0.26*	35.2	51.4	57.7	28.2	50.2	61.2
Fractal (1) (1)	*1.71*	33.0	55.6	61.1	36.7	62.7	72.1
Fractal (2) (2)	*61.35*	75.2	85.9	89.0	69.9	87.8	92.4
Laws	*10.7*	60.6	81.7	88.0	44.9	68.9	77.9
Fourier coefficients	*3.65*	69.7	87.3	91.1	50.8	70.5	79.2
Chen	*41.58*	67.4	86.1	90	60.0	86.1	91.8
Sun et al.	*0.66*	56.3	85.8	92.4	17.1	39.7	49.1
Pikaz et al.	*1.47*	61.3	83.6	90.4	30.8	52.3	61.9
Gabor	*18.14*	67.4	87.1	92.0	64.5	82.5	88.2
Markov	*1.97*	45,2	69.1	76.3	33.8	56.8	67.4
Dapeng	*0.53*	74.4	81.5	95.5	43.8	63.0	70.5
Amadasun	*0.25*	67.4	89.8	94.8	40.8	67.2	77.0
Mao et al.	*2.10*	44.2	69.7	77.8	42.9	64.8	74.9
Amelung	*1.87*	84.2	96.8	98.6	46.2	70.4	80.1
Color							
Average Color Values (RGB)		62.7	84.7	89.5	37.6	58.2	69,6
Shape							
Gradient method	*0.19*	58.2	76.2	83.0	66.9	86.4	90.0
Symmetry method	*0.38*	58.4	77.8	85.3	64.8	86.4	89.7
Central moments	*0.18*	26.8	50.0	61.1	21.9	56.9	75.8
Area		11.2	38.1	54.4	25.7	69.2	88.2
Width and height		35.7	67.8	78.7	58.9	90.6	94.3
Perimeter		3.4	14.6	27.4	20.2	67.7	85.3
Waddel disk diameter		11.0	37.8	54.4	25.7	69.2	88.1
Hydraulic radius		3.6	14.1	26.0	14.4	55.3	72.7
Heywood circularity		2.1	8.6	13.2	7.3	34.3	55.9
Ellipse axes		22.5	46.8	59.9	60.3	90.8	94.4
Ellipse ratio		2.1	8.6	13.2	7.3	33.5	55.3
Rectangle width and height		2.8	8.8	13.6	16.6	49.8	67.8
Rectangle ratio		1.8	8.2	13.5	9.4	41.7	60.6
Compactness		1.7	7.5	11.9	4.4	20.8	33.5
Number of holes		1.0	2.7	5.2	1.4	6.8	10.1
Area of holes		2.3	8.4	15.2	4.5	17.5	25.8
Number of chords X and Y		12.6	35.8	47.5	46.7	84.3	91.8
Maximal chords X and Y		41.0	71.4	81.2	61.6	91.5	95.9
Maximum intercept		1.8	10.6	16.6	12.8	38.4	56.8
Feature groups							
Textural features	*150.48*	89.5	98.3	99.4	68.7	88.2	92.3
Shape features		58.6	77.8	82.9	44.6	71.0	79.3
All features		87.7	96.6	98.1	69.2	88.8	92.3

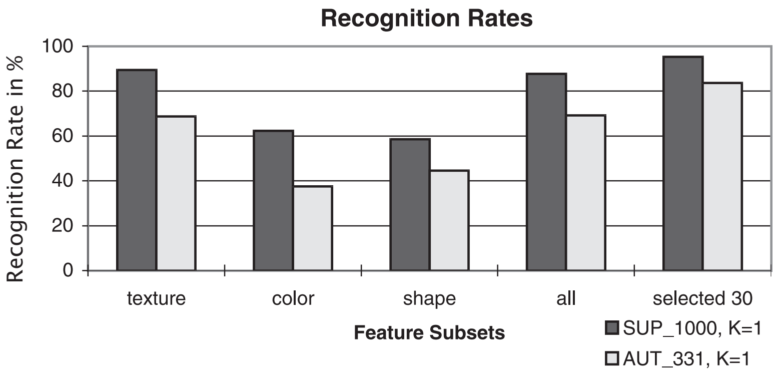


Figure 13.10: *Recognition results for different feature sets: While unselected feature sets do not yield satisfactory results, an automatic selection of features can improve performance considerably.*

methods. Future developments in this field will further increase frame rates, while at the same time include full color processing. For data evaluation in FastCam systems, multiprocessor solutions will provide cheap and scalable computing power.

For object identification in environments with large numbers of object classes, we have shown that feature selection in combination with a variety of features being implemented is an adequate tool to solve recognition tasks. Our next activities in that area will include additional benchmarking parameters into the selection process, such as feature extraction time. In the future, it may be sensible to refer to background knowledge in order to improve the results further .

Special thanks go to all members of the intelligent system group at the Fraunhofer Institute for Integrated Circuits in Erlangen for their continuing support, especially to Steffen Steinhoff, Roland Weigelt, and Christian Küblbeck, and to Siegfried Fößel for his work on the FastCam hardware. Parts of this work have been supported by the Bavarian "Hans-Zehetmair-Habilitationsförderpreis."

13.5 References

[1] Ministerium für Wirtschaft des Landes Nordrhein-Westfalen, (1996). *Produkte und Dienstleistungen für die Bildverarbeitung—Stand und Trends.* Düsseldorf.

[2] Dieckmann, U., Plankensteiner, P., and Wagner, T., (1995). Multi-sensory pattern analysis for person identification with synergetic computers. In *Proceedings of the International Workshop on Automatic Face and Gesture Recognition*, pp. 368–371. Zurich/Switzerland.

[3] Frischholz, R. W. and Spinnler, K. P., (1993). A class of algorithms for real-time subpixel registration. In *Europto Conference*. Munich.

[4] Dongarra, J., (1997). *Linpack benchmark in Java*. Technical Report, http://www.netlib.org/benchmark/linpackjava/.

14 Character Recognition in Industrial and Traffic Control Applications

Reinhard Koy-Oberthür, Thomas Münsterer, and Songyan Sun

VITRONIC Dr.-Ing. Stein Bildverarbeitungssysteme GmbH
Wiesbaden, Germany

14.1 Introduction

The term identification denotes the unique recognition or *classification* of individual objects from a set of objects. In image processing a distinction is made between the so-called direct and indirect identification. Direct identification means the recognition based on directly accessible features of the objects like shape or color. Indirect identification is based on a coding applied to the object. In the following the term identification is always used in the latter sense where the identification takes place by reading the code.

Identification in industrial material flow or in traffic applications fulfills different tasks. In addition to the tracking or distribution of goods it is used for production control and quality management as well

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

as security matters. In certain production processes a combination of marking systems that apply the coding to the object and identification systems that ensure the readability of the codings are used. This is used to ensure identification of the codings in further process steps; therefore, coding verification and identification are closely linked.

To find an optimum solution for an identification application all coding possibilities as well as the coding or marking processes have to be taken into account. Therefore the following sections will deal with these aspects before the concepts of image processing and a character recognition system are discussed. The examples taken from actual industrial installations will illustrate the different fields of applications.

14.2 Codings

There is a multitude of possible forms of coding for technical systems. Nevertheless plain writing and bar codes are the most commonly used, with each form having its specific advantages and disadvantages.

The major advantage of plain writing is obviously the ability to be read and written by human observers. Image processing techniques allow imitation of the human ability to read characters within certain limits. Therefore a change from human spot-checking to a completely automated inspection system can be done with moderate effort. Frequently plain writing is advantageous in that existing processes can be automated without changing the marking system or the process itself. On the other hand, the fact that these writings can be done without defining conditions often leads to significant technical challenges.

To achieve a highly reliable process, the choice of font is crucial. This can be easily seen by looking at the distinctiveness of characters of different fonts:

1 2 3 4 5 6 7 8 9 0 O I B	Lucida bright
1 2 3 4 5 6 7 8 9 0 O I B	Lucida sans serif
1 2 3 4 5 6 7 8 9 0 O I B	Courier
1 2 3 4 5 6 7 8 9 0 O I B	OCR-A
1 2 3 4 5 6 7 8 9 0 O I B	OCR-B

The digits 3 and 8 or 8 and 9 in most fonts are almost identical except for openings in the character. The example of the Courier font shows that these partial identities, which can lead to mistakes in the presence of distortions, can be avoided. Fonts like OCR-A or OCR-B are further optimized for automated readability. In addition, the introduction of a checksum can reduce almost to zero the probability of mistakes or mixed-up characters.

14.3 Code generation and code control

The markings can be applied using a multitude of techniques. The following list shows the most important techniques used in industrial process control.

- printing with inkjet on labels or directly on metal surfaces
- thermotransfer printing of labels
- needle engraved markings or embossing die on metal or other materials
- characters applied by spraying dye at a template (e. g., railroad cars)
- hot spray or metal powder markings in the steel industry
- laser engraved markings (e. g., in wood, glass, metal)
- raised characters in casting molds
- punching characters, for example, in type plates
- different sorts of printing on different materials
- *handwritten characters*

The multitude of different marking processes and the individual history of each part to be identified results in a wide range of writing qualities. In addition to this, the layout of writings and the corresponding background structures are manifold.

To ensure readability by a computer system in the course of the production process a first reading is often done right after the marking. Based on the *a priori* knowledge this reading is rather a verification—not an identification. An example of verification is the automated marking of product data on products or packings (for example, in the pharmaceutical industry).

14.4 Functional principle and system setup

A system intended for industrial character recognition should be capable of dealing with the wide variety of different conditions and tasks described in the preceding text. For this reason such a system has to meet the following demands:

- Optimization of image acquisition and illumination to the relevant kind of coding and object surface
- Imaging of contrasting and noncontrasting two- and three-dimensional codings that can either be at rest or moving
- automated search for codings on structured backgrounds and compensation of positioning tolerances

- adaptation to changes in size and rotation angle of the writing (i. e., size and rotation invariance of reading)
- separation of characters in spite of distortions and interconnections
- possibility of adapting the classification technique performance to the degree of distortion of the characters
- capability of reading different printed fonts or handwritten characters
- adaptation of computational effort and power to the given time limits
- communication with different host computers or PLCs using different protocols

The forementioned profile asks for a flexible hardware concept with respect to image acquisition, computational power and the communication possibilities of the system. In addition, such a system needs a higher-level control system as well as the possibility of changing the various working steps of the system, that is, an adequate teach-in and a graphical user interface are required.

14.4.1 Architecture of an industrial character recognition system

Keeping the preceding list in mind the image processing system VI-CODE was developed. The functional architecture of the system is shown in Fig. 14.1. In this architecture reading plain writing is realized as a process of information reduction starting from the image to the location of the writing, to the line of writing, to a single character and finally to the identification of this character. The essential characteristic of this architecture includes knowledge-based methods on every level where there is *a priori knowledge*. This *a priori* knowledge of size and positioning of the characters is, for example, used in several levels of the processing hierarchy. For classification purposes the available knowledge of the possible contents is used in the form of automatically generated hypotheses.

In this process the sequence of the different steps is not fixed within the hierarchical information reduction scheme. Rather an evaluation of the results takes place at every process level that determines the progress of the reading process. Therefore processing steps can be repeated with changed parameters in a recursive way. Result evaluation also uses *a priori* knowledge. This *a priori* knowledge is given to the system by the teach-in process in the form of a changeable information basis. This information basis contains information about the system configuration, the process sequence and process parameters, which are usually set for each application by the system developer. Also the fonts and the decision thresholds for the classification are part of this user-

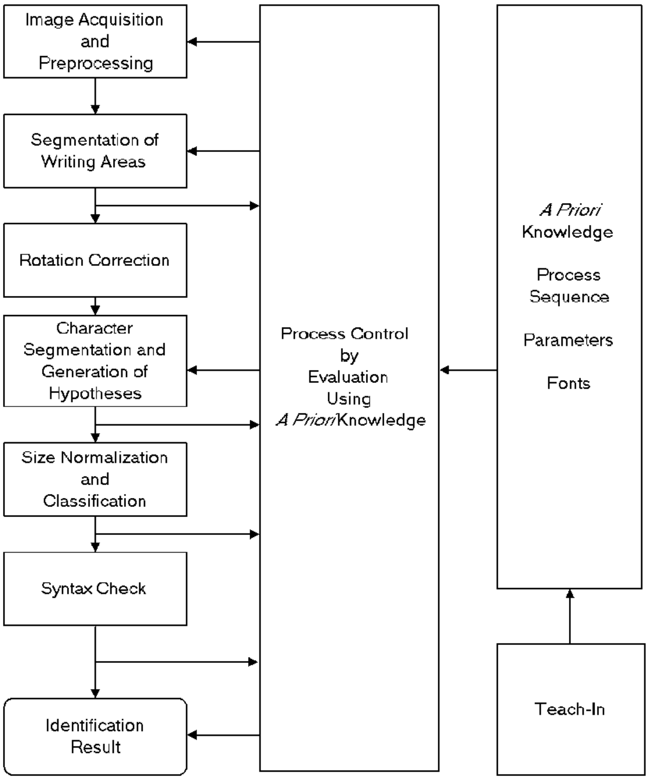


Image Acquisition and Preprocessing
Segmentation of Writing Areas
Rotation Correction
Character Segmentation and Generation of Hypotheses
Size Normalization and Classification
Syntax Check
Identification Result
Process Control by Evaluation Using *A Priori* Knowledge
A Priori Knowledge
Process Sequence
Parameters
Fonts
Teach-In

Figure 14.1: *Functional architecture of the optical character recognition and image processing system VICODE.*

given information basis. For giving this information to the system the user is guided through a comfortable graphical user interface.

An additional and essential feature is the ability to deal with multiple fonts. Depending on the application different variants of a single character can be taught as a single font or the different variants are handled separately as different fonts. The system uses a font classification that supports an automated selection of the relevant font.

Figure 14.1 illustrates that several processing steps are necessary for a reliable classification in a broad application range. Even the image acquisition and illumination can be influenced depending on the actual image content. For preprocessing of the acquired images different geometrical transformations and linear as well as morphological filter operations for distortion suppression or to compensate for inhomogeneous illumination can be activated.

The separation, that is, the precise location and division of every single character is of major importance. Separation is a good example

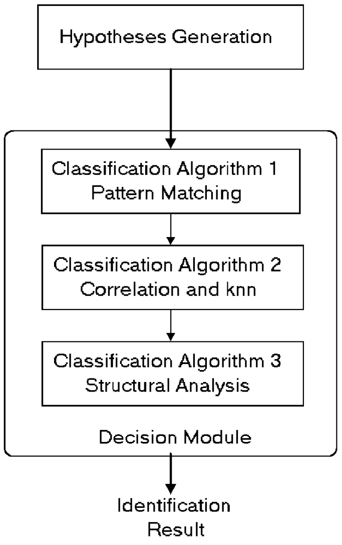


Figure 14.2: *Structure of the 3-level classification scheme.*

for a knowledge-based recursive process that may be performed several times with changed parameters depending on the result and *a priori* knowledge. The separation algorithms can be a linear or a nonlinear histogram function as well as a correlation function.

An important system feature is the size and rotation invariance. These invariances are realized as independent functions that take place before classification. This means that the actual character recognition is done with normalized characters.

For the classification of characters, the Optical Character Recognition (OCR), a multitude of algorithms have been developed over the last 30 yr or so. To combine different classification algorithms with excluding characteristics we use a multiclassifier approach with a three-level classification scheme (see Fig. 14.2).

The classificators are hierarchically organized with respect to hypothesis reduction. The results of each level are combined to a final result in a separate decision module. Each classification module has a quality measure of its result and a sorted hypotheses set. These are evaluated by the decision module and according to this evaluation an additional classification module is activated or the final result is calculated. This leads to an optimal recognition rate with minimal calculational costs. The classification schemes used here are based on work done by Fukunaga [1], Kahan et al. [2]. Similar classification schemes are used for object classification in Volume 2, Chapter 26.3 and, more generally, in Volume 2, Chapters 26 and 27.

The classification algorithm 1 in Fig. 14.2 is a pattern-matching algorithm, that is, a direct comparison of the normalized character with a taught mask. For undistorted characters this very fast method is successful. If it fails a complete correlation is performed. This algorithm can be enhanced by using a next neighbor decision module, which allows lower thresholds to be used. Classification algorithm 3 performs a structural analysis of the characters. For this the skeleton of the character is calculated as an abstract representation ([see 3, 4] and also Volume 2, Sections 21.3.9 and 21.3.10). This skeleton is transformed into straight lines and circle segments. These segments are finally described as fuzzy shapes to cover variable shapes of the same character [see 5]. The classification is based on these fuzzy features.

14.4.2 System hardware and process interfacing

Demands concerning time for reading range from 5 characters/s to 200 characters/s depending on the application. These different demands can be fulfilled by a modular hardware concept. The system is based on PCs with Pentium processors built as a single or multiple processor system. Part of the modular hardware concept is the configurable image acquisition that can use different line array or matrix cameras or even 3-D image acquisition systems. For the latter case, which can be used for reading embossed or punched characters, a light stripe scanner for generating a 3-D image is used. Line array cameras are used if large areas have to be covered with high resolution. In these cases the line array camera is either scanned across the object or the object passes by the camera at a constant speed. This technique allows for image sizes of more than 7000×7000 pixels. For process interfacing standardized digital inputs and outputs as well as serial interfaces using different protocols are available.

14.5 Examples of applications

14.5.1 Identification in the automobile industry

Automated identification systems are widely used for quality management of assembly processes in the automobile industry. As an example, the task described here is to read needle engraved characters on the surface of fluid drive castings. These castings are marked right after manufacturing to ensure that castings with different interior structures but equal sizes can be distinguished. At the assembly line the 6 digit marking on the castings is to be read and compared to the preset identification number. The steel surface ranges from mirror-like to pale gray and can be scratched or spotted. The markings are applied by up to three different needle markers with different fonts.

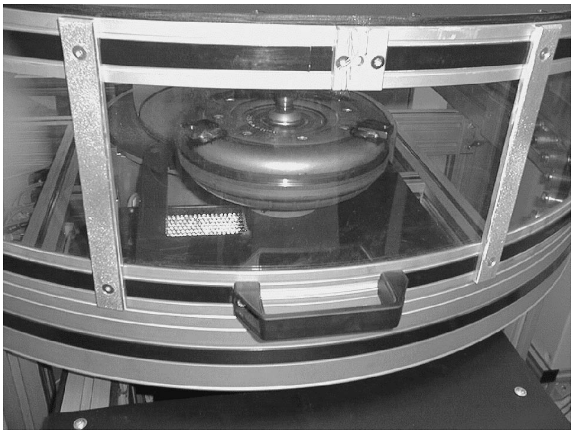


Figure 14.3: *Side view of an identification in progress. The casting is already lifted up from the work piece slide. The flashing LED lamps can be seen through the window in the work piece slide.*

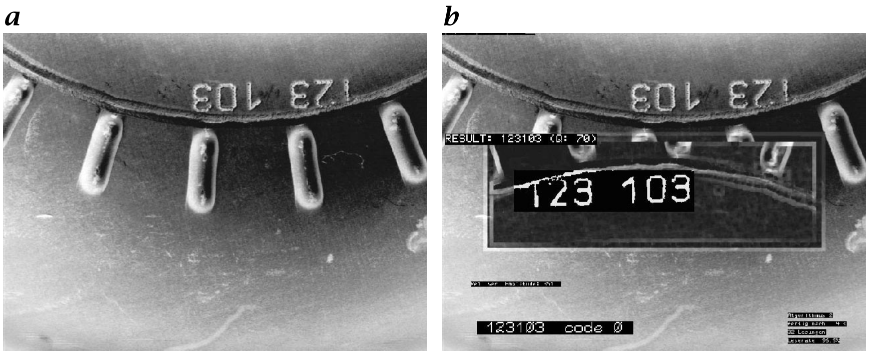


Figure 14.4: ***a*** *Original marking on a casting found on the rotating casting. The digits are distorted by the trace of a defect traction wheel.* ***b*** *Identified number of this casting.*

In operation the assembled parts sit on work piece slides that are separated and stopped at the identification site. The parts are then lifted from the assembly line and rotated with a constant speed of 5 rpm. The part is illuminated under a 30° angle from below with two red LED arrays that are flashed to suppress motion blur. The camera looks up vertically through a window in the work piece slide (see Fig. 14.3). Images of the rotating part are acquired in real time, that is, 25 frames/s. Also a character-seeking process takes place in real-time. A fast algorithm for finding a line of digits is to look for structured edges, that is, for clusters of pairs of edges. This algorithm uses a standard recursive edge detection algorithm that is adapted to this task

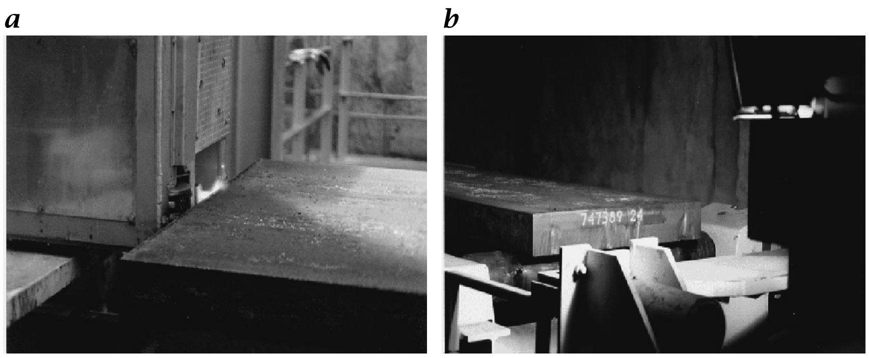


Figure 14.5: ***a*** *Marking of the red-hot steel slabs: The metal power marker can be seen on the left-hand side of the image.* ***b*** *Steel slab with marking being identified. the camera is enclosed in the housing on the right hand-side of the image. The lamp is mounted on top of the housing.*

[see 6] (see also Volume 2, Chapter 11.4.1). As soon as the marking is found, 5 consecutive images are stored and character recognition takes place on these 5 images. The time limit for a complete circle including lift up, rotation, character search, reading, and let down is 25 s. For an optimal result even with scratched or distorted characters reading takes place with up to 3 different parameter sets using between 1 and 6 s. The identification rate is better than 98% for all castings.

Figure 14.4 shows an example of the imaged markings found in real-time (left) and the binarized and read digits on the right. If a marking cannot be read by the system, which is usually the case for heavily distorted or scratched characters, the assembly line is blocked and the user has to type in the code he sees on the monitor. The assembly line is also blocked if a code is read that is different from the preset one, that is, when a false part is on the assembly line.

14.5.2 Identification in steel plants

In steel plants the goal of identification is total quality management, with the superior intention of ensuring constant quality for the final customer. The parts are marked with characters because they can also be read by humans. The marking is either applied in the hot area (800-900 °C) by spraying metal powder characters or in the cold area (meaning areas with temperatures below 400 °C) by the dot paint method.

The site of the system described here is the continuous casting machine of a German steel plant (see Fig. 14.5). After the chill form the steel is moved over a roller table. At this roller table parts are cut from the billet, the so-called slabs. This cutting process is done by a gas cutting installation that is moving along with the steel billet.

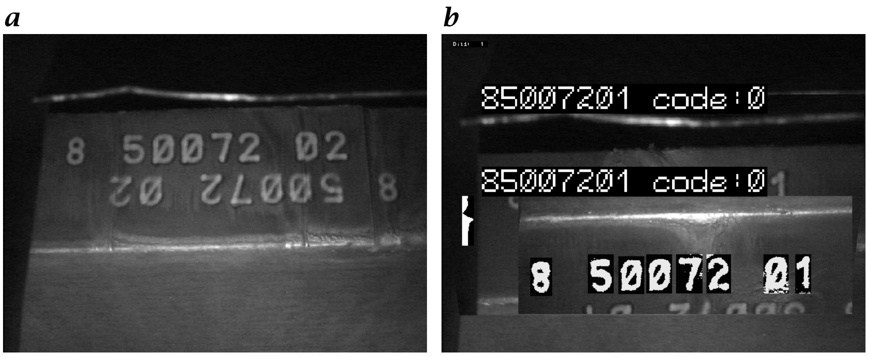


Figure 14.6: ***a*** *Original image of a marked steel slab seen from the camera,* ***b*** *processed (binarized) and identified image.*

After being cut the slabs are marked with an individual ID number. To ensure subsequent identification of the slabs a first identification or verification takes place right after the marking. At this point the surface temperature of the steel at the cut edge is approximately 800-900 °C. In addition to the heat of the hot slabs the characteristic radiation of the red hot steel is of major importance to any image processing system. The characteristic radiation must not interfere with identification quality. This problem is solved by using a special illumination in the green spectral range with the use of cut-off filters and additional hot mirrors to protect these filters and the optical system. The distance from the camera to the steel slabs is approximately 1 m, the imaged size is approximately 0.85 m. The camera is enclosed in a water-cooled steel housing.

In addition to the environmental conditions the mixed marking quality and especially the structured background together with outdoor and daylight operation is a major challenge here. The cut edge that was produced by the gas cutting installation shows irregularities, such as thick grooves. Figure 14.6 shows the cut edge of a steel slab seen from the camera system. An identification rate of greater than 98 % was achieved.

14.5.3 Identification of license plates

The application described here and in Fig. 14.7 is installed at a waste disposal site. The entrance and exit areas are monitored for goods movement and general security. At the entrance and exit area the trucks have to stop at a weighing station. The driver has to identify the customer through an ID card with a magnetic stripe. The position of the truck is known up to ±1 m. The card reader gives a trigger signal that starts the identification process of the truck, that is, the identification of the license plate. The image processing system starts the image ac-

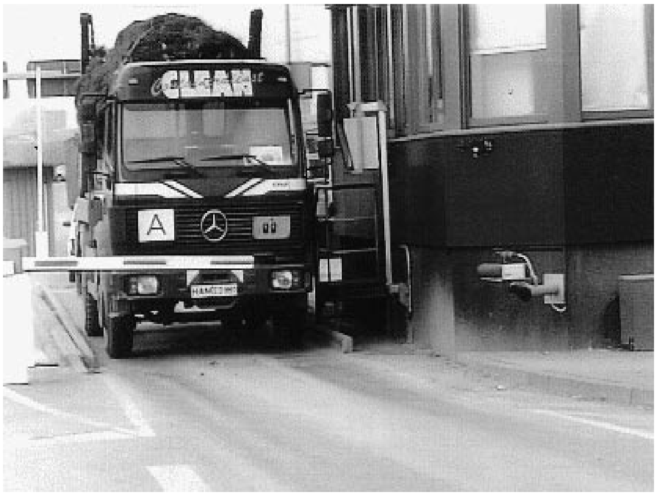


Figure 14.7: *The OCR system VICODE identifies the license plate of the garbage truck at the entrance. Simultaneously the truck is weighed and the driver has to identify himself. The camera system can be seen on the right-hand side of the image.*

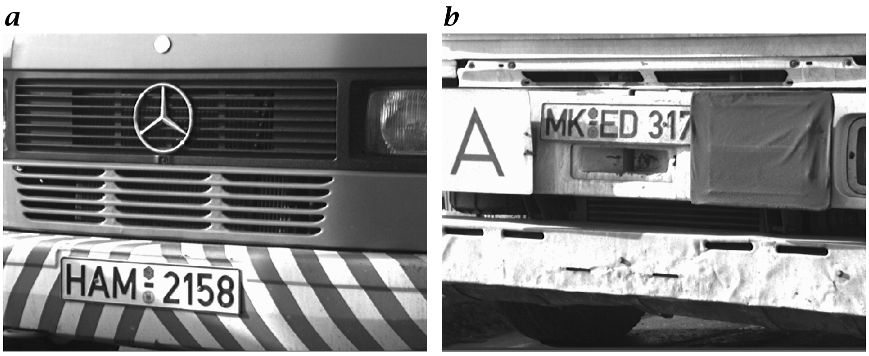


Figure 14.8: *Typical examples of license plates imaged by the camera system. Both images show the problems for an automated search and identification algorithm.*

quisition that is adapted to outdoor use by applying an infrared lamp with adequate color filters. The range of vision is 0.9 by 1.2 m. The distance between the camera and the license plate is 3.0 m at an angle of approximately 35°. The described setup also takes care of the process inherent position and distance tolerances. In addition to the standard process described in Section 14.4.1 an additional search algorithm is used to find the license plates in the acquired image. Similar to the algorithm designed for detecting the presence of characters (see Sec-

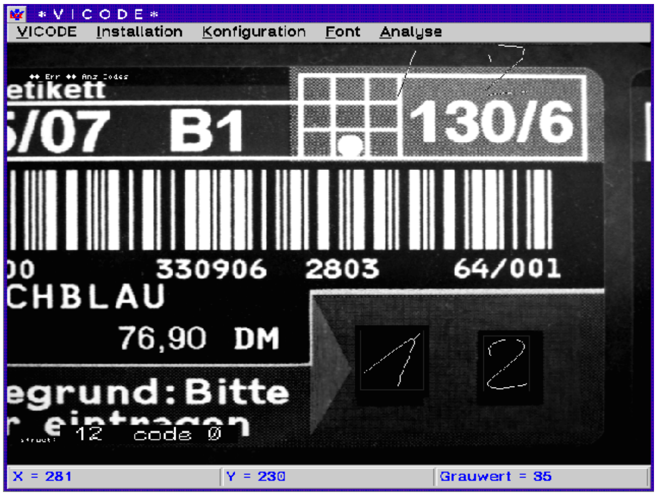


Figure 14.9: Example of a package label. The customer is supposed to write the reason for returning the goods into the two white boxes. In practice, many customers do not write the numbers into these boxes. In these cases the image processing system has to separate the handwriting from the background colors.

tion 14.5.1) this algorithm is based on the presence of structured edge pairs. Using the constraint that characters have a certain stroke width and structure this algorithm is capable of finding license plates on a randomly structured vehicle front. In addition, special algorithms are used to cope with typical disturbances like screws, dirt and tags. The multifont capability allows for the processing of older German license plates as well as the new so-called Euro license plates. Additional fonts could be taught and handled as well (see Fig. 14.8).

Due to the required time limit of several seconds a single computer system can perform both the entrance and exit identification but simulates two independent identification systems towards the host computer. After the automated identification of the vehicle data at the exit weighing station a deduction can be automatically printed for registered customers.

The identification rate is limited by damaged or extremely dirty license plates. The system also cannot read license plates that are not properly fixed at the vehicle front (e. g., behind the window). Therefore a standard procedure that takes care of nonreadable license plates was included. The image of the video camera is displayed at the monitor in the central office for manual input. Based on the experiences at this waste disposal site an identification rate of 95 % under "real world conditions" can be reached.

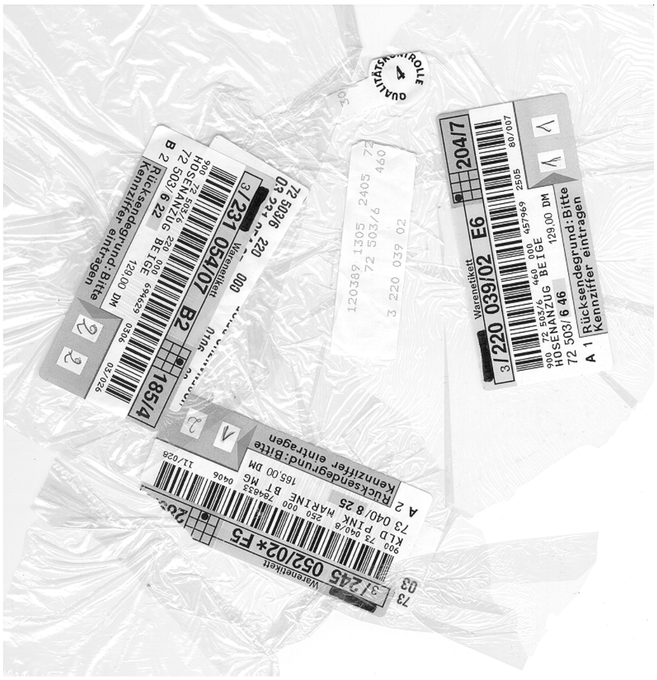


Figure 14.10: Example of package labels, which are processed in any given orientation at a conveyor belt speed of 0.5 m/s.

14.5.4 Identification of mailing labels for the mail order industry

Mail order houses guarantee their customers a right of return in case they dislike the ordered goods. This leads to a large number of returned goods that have to be handled by the mail order companies. These returned goods have to be identified and returned to stock. The reason for returning the goods has to be given by the customer—coded as a handwritten two-digit number on the return label (see Fig. 14.9). This information is handled for statistical reasons.

In this application differently packed returned goods are transported past the identification system. The packages are placed by hand on a 0.6 m wide conveyor belt that has a speed of 0.5 m/s. The size of the goods varies from single packing labels to larger parcels with sizes of up to 500 mm × 400 mm × 300 mm. The parcels are labeled with a standardized return label that has a 22-digit barcode and a field for two handwritten digits. The shape of the returned goods is only limited by a maximum size. Therefore the labels can have any orientation and can even be corrugated (see Fig. 14.10). In addition label orientation of ±50° against the horizontal is also allowed here.

To fulfill this task a system combining a barcode and *OCR reading* software was set up. The sensor unit was designed so as to cover the full height of 300 mm and the full width of 450 mm. This was done by 8 matrix cameras and a Xenon flash illumination to minimize motion blur.

After the startup the complete imaged sector is searched for labels without an extra external trigger. For this search the mapped and calibrated images of all sensors are digitized in real time and a search algorithm that searches the barcode is activated. If a single image contains only part of the barcode the information is put together from the adjoining images.

Starting from the *barcode* position the possible image sectors that contain the handwritten digits are calculated. The identification process described in Section 14.4.1 was extended by classification modules for classifying handwritten characters by a structural analysis and a so-called *knn* classifier [see 7].

The automatic handling of returns considerably increases the capacity. Due to the program's sturdiness regarding the quality of labels the use of high-quality labels is not required.

14.6 References

[1] Fukunaga, K., (1972). *Introduction to Statistical Pattern Recognition*. Boston: Academic Press.

[2] Kahan, S., Pavlidis, T., and Baird, H. S., (1987). On the recognition of printed characters of any font and size. *IEEE Trans. PAMI*, **9**:274–288.

[3] Zamperoni, P., (1989). *Methoden der digitalen Bildsignalverarbeitung*. Braunschweig: Vieweg.

[4] Ballard, D. H. and Brown, C. M., (1982). *Computer Vision*. Englewood Cliffs, NJ: Prentice Hall.

[5] Zimmermann, H.-J., (1991). *Fuzzy Set Theory and its Applications*. Boston: Kluwer Academic Publ.

[6] Deriche, R., (1990). Fast algorithm for low-level vision. *IEEE Trans. PAMI*, **12**:78–87.

[7] Nagy, G., (1982). Optical character recognition—theory and practice. In *Handbook of Statistics*, P. Krishnaiah and L. Kanal, eds., Vol. 2, pp. 621–649. Amsterdam: North-Holland Publishing Company.

15 Motion Tracking

Robert Frischholz

Mikromak GmbH, Erlangen, Germany

15.1 Introduction

The quantitative acquisition of the movement of certain objects in a *video* sequence, *motion analysis*, is required in many fields. In sports, the training progress of athletes must be examined quantitatively; in medicine, people's progress in rehabilitation must be measured especially for gait analysis; in industry, car crash tests must be recorded and analyzed exactly.

A motion analysis system, therefore, usually consists of a video camera, a digitizing unit, and a PC with a certain software to acquire object locations for each image. For the analysis of fast events, digital *high-speed cameras* are often used.

Although analysis of the video sequences is often still done manually, there are several approaches for automatic tracking algorithms.

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

The task of these algorithms is the automatic recognition of certain objects within the image, the determination of their locations, and the tracking of those objects along the complete sequence. Typical approaches for this task use special markers applied to the moving objects. Often circular-shaped markers, black or white or reflective, are used. Feature extraction and classification then detects the position of each marker in each image.

After an overview of existing techniques for motion tracking, an automatic motion analysis system is presented, which overcomes the restrictions of using special markers. The main components, including adaptive *template-matching* based tracking, neural net approximation, and 3-D calibration, are explained to show how combined image processing and pattern analysis algorithms form a complete motion analysis system.

15.1.1 Definition of terms

The task of visual motion tracking, also often called motion analysis, is the spatial and temporal acquisition of moving objects in image sequences. Each object position is determined in either abstract coordinates (pixels) or in physical units (meters). All locations of an object in successive images of the whole sequence make up the so-called trajectory. Therefore, a *trajectory* is a discrete time function. Figure 15.1 gives an example of an image sequence and the corresponding trajectory.

Extracting trajectories of moving objects in image sequences is in principle possible using two different approaches: manual or automatic. For both methods, computers can increase performance as well as accuracy.

15.1.2 Requirements of motion analysis

The analysis of moving objects is required in many fields, with specific and varying constraints. Here is an overview over the main fields of application:

Automotive industry. In car crash tests, both the movement and deformations on dummies and the car parts itself need to be analyzed. As the experiments are taken under controlled conditions, it is no problem to apply markers on the parts that are of interest. Also the time for analysis is not critical and can take up to several days, whereas the precision of object locations and computed accelerations is very important. High-speed cameras must be used to sufficiently record the fast movements. Typically, 1000 frames/s or more is used.

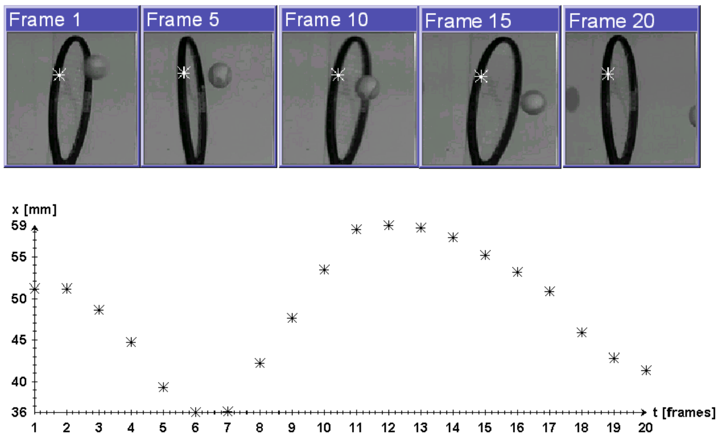


Figure 15.1: *Example of an image sequence (5 sample frames out of a total of 20, for full sequence, see `/movies/15/test.mov`) of a tennis ball hitting a racket. The horizontal position of the marked point on the racket over time defines the trajectory. Here, it visualizes the horizontal swing of the racket.*

Machine part inspection. To investigate the behavior of machines in a production line, it is normally not possible to apply any markers. As already mentioned, requirements are very high for precision and recording speed.

Sports equipment. To investigate the material of, for example, rackets or sport shoes, it is often possible to attach markers to the equipment. What differs from the forementioned industrial applications is the requirement of fast analysis—often, the many test-rows that are taken require fast results. Precision requirements are sometimes very high, as measuring, for example, angle velocities on the human foot requires very high accuracy. Recording speed is usually in the middle high-speed camera level, about 200 frames/s is mostly sufficient.

Athlete training. For the comparison and improvement of athletes, motion trajectories of certain exercises are gathered and analyzed. Here it is often not possible to apply markers to the athletes in competition. Therefore, and because of the high deformability of human body parts, object locations sometimes vary with an uncertainty in the range of centimeters. Nevertheless, higher precision would be preferred. Human movements are often recorded with standard video cameras or with high-speed cameras of 200–500 frames/s. There are, of course, certain movements like a golf swing, that require an even higher frequency.

Medicine. The main medical application, clinical gait analysis, is usually done in a laboratory under controlled conditions. Markers can be applied to the patient, who is usually observed by two to four video cameras for 3-D measurements. The time requirements to get the results vary from close to real-time demands of up to several hours, if a complete clinical report is intended. Another medical application, the analysis of vocal cord vibrations, differs in the required sampling frequency—here, usually several thousand frames/s is needed.

To summarize, there is a wide variety of applications for motion tracking. Generally said, the higher the frequency, the more data needs to be analyzed, and the higher are the performance requirements for the motion analysis system.

15.1.3 Methods of motion analysis

Today, the most common method for motion analysis extracts the object locations manually for each individual image frame. A typical working place consists of a remote controllable VCR connected to a computer with a video overlay board. The VCR is controlled by software, which stops the videotape at the desired video frame, and allows for selecting and clicking point locations on the overlay video. After all locations have been edited, the controller software moves the VCR to the next video frame. After this manual point acquisition (often confusingly called "digitizing," but this does not necessarily imply that the video images are converted to digital video), the trajectories are digitally stored and ready for further analysis (e. g., calculating velocities, accelerations and frequencies). As an example: Until the late 1980s, the analysis of vocal cord vibrations was done manually. Childers [1], Tanabe et al. [2] marked the tracking points on the vocal folds using a light-pen on a computer screen. This task took about one hour for approximately 100 image frames.

Manual motion analysis benefits from an experienced operator, who can identify locations from experience under circumstances where automatic image processing fails. On the other hand, this method is very time consuming and not very precise (although the software could be able to use tools such as digital zooms for precise point location, the operator will normally edit points at pixel accuracy only).

Advances in image processing have led to automatic approaches for motion analysis. The automatic methods can typically be separated into online (real-time) and offline (after recording) tracking. Both of them usually require markers on the objects to be tracked. Markers can be passive or active by reflecting or emitting light, they are usually round, sometimes 3-D or even colored. In Fig. 15.2 several typical

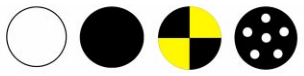


Figure 15.2: *Typical markers: White or black circles for fast detection; more complex shaped markers for offline analysis.*

markers are depicted. For real-time requirements, simple segmentation algorithms must be chosen to acquire the center of the markers, and, therefore, usually black or white markers are used. For offline analysis, more complex markers and algorithms can be used for the task of object detection.

Marker-oriented methods can only be used when the objects to be tracked are suitable for marker applicants. For some special tasks, especially with nonrigid motion and deformation, like the human heart, air bags, clouds, these methods are under normal conditions not applicable. Therefore, recent publications tend to concentrate on qualitative motion detection, trying no longer to determine only the movement of a single object point, but to calculate motion fields of compound objects. See Volume 2, Chapters 14 and 15.

Although those methods are designed to deal with very complex scenes, they are only partially valuable for the typical motion analysis problem—which involves determining the *exact* movement of a certain object point.

15.2 Flexible automatic motion tracking

Only recently, scientific image processing know-how has moved from static scene analysis to more complex motion analysis applications. By means of pattern recognition algorithms and neural net technology, the task of motion analysis has been made more flexible and powerful. In the following, our motion tracking system is described, by referring to the most common problems in motion analysis, and their new solutions in the presented system.

15.2.1 Object tracking

Instead of using black/white or reflective markers, the flexible technique of *template matching* [3] is used. The procedure of template matching is basically as follows: A small area surrounding the point to be tracked is used as the template. This template is then searched for in the next image frame by use of correlation techniques. The location with the highest correlation result is the best match between template and image (see Fig. 15.3).

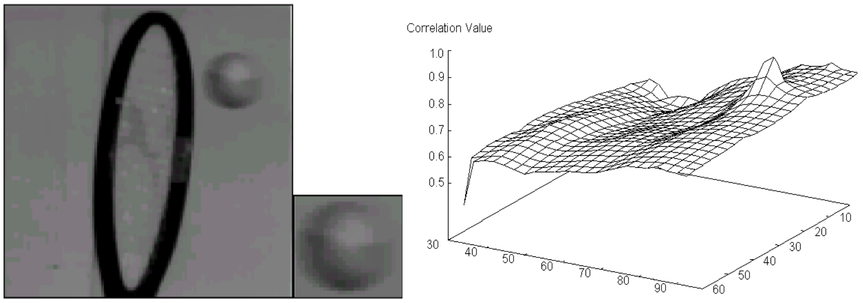


Figure 15.3: *Original image (left) showing a ball and a tennis racket; template (middle) used for correlation; correlation surface (right)—highest value indicates best matching position.*

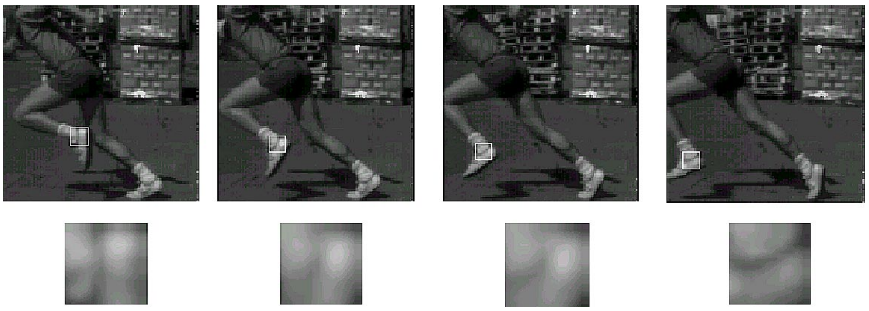


Figure 15.4: *Example sequence with the corresponding adapted templates. The rectangular area of the respective templates is drawn as a white rectangle in the images.*

Template matching is a very flexible and powerful method for tracking objects with small distortions from one frame to the next. By adapting the templates along the image sequence, even larger deformations can be tracked [4]. The adaptation used here consists of a selection of basically three adaptation rules:

- If there is a large change between the original and the new template location, the new template location is selected. In this case, templates are completely exchanged by their new shape.
- If there are small changes between original and new locations, an averaged version of old and new template is computed and used as the new adapted template. By doing this, small derivations due to noise are averaged, thus increasing the tracking stability against noise.
- If there are only minor changes between original and new locations, the old template is used. This is very important for objects

with translations by amounts smaller than one pixel: An adaptation would lose the subpixel shift information.

Figure 15.4 shows the adaptation phases of a template of a rotating object: The first template is the original object subimage, the template is then adapted to the (rotated) subimage in the second image, and so on. By using this kind of adaptation, even large rotations can be tracked in an image sequence. The necessary precondition is, of course, that the changes from one frame to the next are small enough for the template matching to still locate the object correctly.

15.2.2 Precision

When using image processing algorithms, point locations can be determined within *subpixel* accuracy [5]. Subpixel techniques can, by means of interpixel interpolation and subsampling, improve the object detection location to fractions of a pixel. For the application in motion tracking, an average location accuracy of 0.05 pixel has been reported [5]. This enhancement in location precision is especially useful when calculating derivatives from the trajectories, because then the typical noise induced by pixel errors is reduced dramatically.

A simple algorithm for subpixel measurement is the interpolation of the correlation values. Considering only one dimension, the correlation neighborhood can be shown as follows.

H is denoted as the hitpoint, that is, the point with the highest correlation value. Both $H-1$ and $H+1$ must, of course, have lower correlation values. Let the correlation value be f_m at location H, f_l at $H-1$, and f_r at $H+1$. By these 3 points, a parabola is well defined. The maximum of this parabola determines the subpixel hitpoint H_{sub}:

$$H_{sub} = H + \frac{f_r - f_l}{2(2f_m - f_r - f_l)} \tag{15.1}$$

Investigations of this method showed that an average error of about ± 0.13 pixel can be achieved [6].

Instead of a parabolic approximation, the three values f_l, f_m and f_r can be interpreted as bars with a corresponding mass. The center of mass then yields another value for the subpixel point:

$$H_{sub} = H + \frac{f_r - f_l}{f_l + f_m + f_r} \tag{15.2}$$

Here, investigators reported a slightly better average error of about $\pm$ 0.10 pixel [5]. But, when comparing those two methods, an interesting symmetrical behavior can be recognized (see Fig. 15.5), which was proven in [7]. Therefore, the arithmetic mean of both methods yields a much better precision: The average error of the combined method

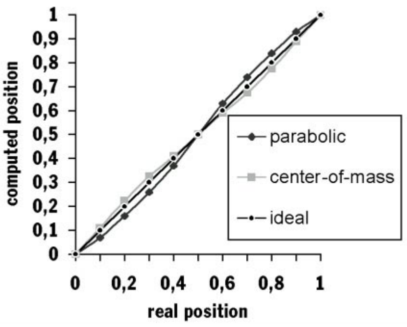


Figure 15.5: *Comparison of parabolic and center-of-mass approximation; for test sequence and test results see* `/movies/15/test.mov` *and* `/images/15`*, respectively.*

is about ± 0.05 pixel, measured both on artificial and real images [7]. In the system described here, this improved method was implemented, thus enhancing pixel accuracy of object tracking by a factor of 20.

15.2.3 Occlusion

Tracking objects that are partially occluded by another one is not only a problem for automatic algorithms, but even human operators can only judge from their experience the location of an occluded point. For example, in typical gait analysis, one leg occludes the other; even hip positions may be occluded by the patient's hand. An example is given in Fig. 15.6: By tracking the hip, knee and foot in a walking cycle of one second, the trajectories are divided into 6 gaps because of occlusion.

When more than two cameras are used, occluded points can be reconstructed with the knowledge from at least two camera views. But when only up to two cameras are used, the gaps in the trajectories have to be interpolated. The new approach of our system for solving the occlusion problem is by means of neural net approximation.

Neural nets have the ability to learn complex nonlinear functions by just presenting them a set of input and desired output values. While this ability is normally used for class separation in classification tasks, it can be used as well for the approximation of functions.

When a neural net is trained with all values of a trajectory, it builds an approximation of the underlying function. After the training phase, all missing points can be reconstructed [4]. In our system, a standard backpropagation perceptron was used with a simple topology: The input layer consists of one neuron (for the time scale inputs), the hidden layer uses several neurons for functional approximation, and the out-

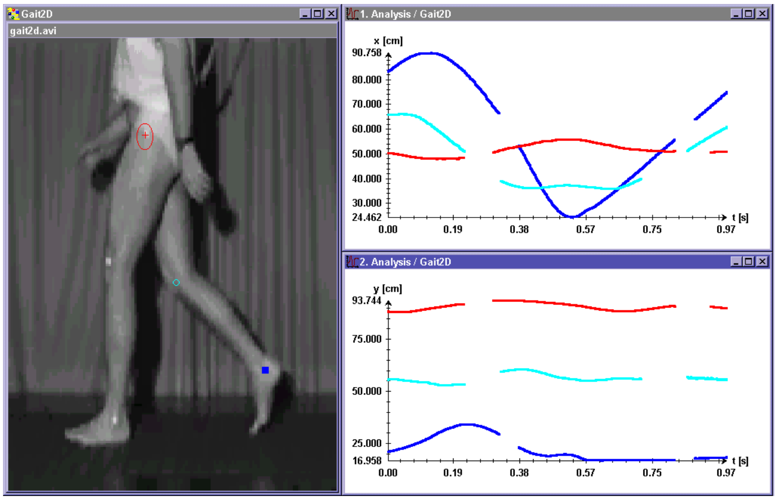


Figure 15.6: *Picture of a gait analysis experiment (left); xt trajectories showing gaps caused by occlusion (upper right); yt trajectories: Hip, knee, and foot shown from top to bottom (lower right).*

put layer again consists of one neuron (for the spatial outputs). The sigmoid function was used in all neurons.

What follows is an example of the learning behavior of such a neural net. From the example of the introduction, a gap was introduced into the trajectory of the tennis racket sequence.

By using a 1–6–1 neural net the time values were presented as input values, and the horizontal coordinates were presented as output values.

After the termination criterion (each single pixel error below 0.1) has been reached, neural net approximation terminates and fills the gap. In the example already mentioned, comparison with the perfect original values showed an average approximation error of about one pixel, which, considering the difficult position of the gap (maximum peak), can be considered as a very good approximation.

Training a neural net is very time consuming. The foregoing example needed 105 s of computation time on a PC Pentium 150 MHz. On the other side, neural net approximation has several advantages:

- No approximation function (polynomial order etc.) has to be selected
- The whole shape of the trajectory is considered
- Complex nonlinear functions can be approximated

These advantages are very useful for practical applications like the gait analysis example of Fig. 15.6: The gaps in the trajectories of the walking cycle can be filled automatically without any user interaction, while considering the different shapes of each individual trajectory. The 12 gaps overall (6 in each x- and y-direction) are interpolated within about three min with our software on a PC Pentium 200 MHz.

15.2.4 Three-dimensional calibration

To obtain 3-D metric values out of the tracked points, at least two cameras must be used. When all camera parameters are known to the system, the 3-D coordinates can be calculated out of a pair of corresponding 2-D point coordinates. Considering standard stereo vision, for every point in the left image a corresponding point in the right image must be found. This correspondence problem is not commonly solvable.

By applying some interaction, however, software can do valuable work in this area. The basic idea of our system is therefore not to solve the correspondence problem with a computer algorithm, but with human help. The rest is done automatically by the software.

In practice this means that—equal as in the 2-D case—all object points to be tracked are edited manually for left and right image. The critical task of finding the proper correspondence is by this way established by the human operator. All other frames of the sequence can now be tracked automatically, following the principles already laid out. Real 3-D world coordinates are computed after tracking by use of the stereo trajectories. To achieve this computation, cameras have to be calibrated.

For flexibility, the calibration method must allow an arbitrary positioning of both cameras. The task of the calibration method is the computation of all internal (focal length, lens distortion) and external (rotation and translation angles) parameters of each camera. To calibrate a camera, usually a calibration model is placed in the area of interest. The precise 3-D measures (so-called “ground truth data”) of the calibration model are known. The camera takes a picture of the calibration model, and the projected x- and y-coordinates of each calibration point are determined. By solving the equations of the mathematical camera model, all camera parameters can be determined.

We implemented the calibration method suggested by Tsai [8]. Calibration models to be used can be both coplanar and noncoplanar. Noncoplanar (volume) models have the advantage of giving more precise calibration data over the coplanar models, which in turn are easier to handle (see the noncoplanar model in Fig. 15.7).

The calibration process, which has to be done only once as long as the cameras are not moved, may take some minutes. After getting all camera parameters, it is easy to calculate the 3-D world coordinates out of the two stereo trajectories.

Camera calibration seems to require much effort. Nevertheless, there are several advantages over restrained stereoscopic methods (e. g.,

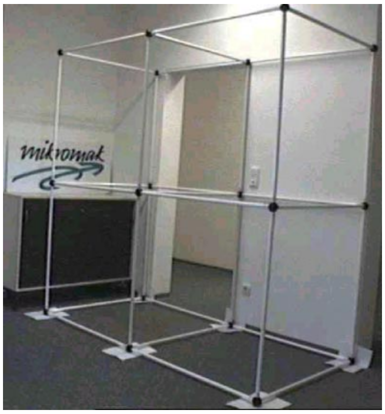


Figure 15.7: *Example of a noncoplanar calibration model; calibration points are the spheres on the stick crosses. This model spans a room of* $2 \times 2 \times 1\,m^3$*, which are the minimum requirements for human gait analysis.*

using two cameras with fixed distance, a common axis and equal focal length):

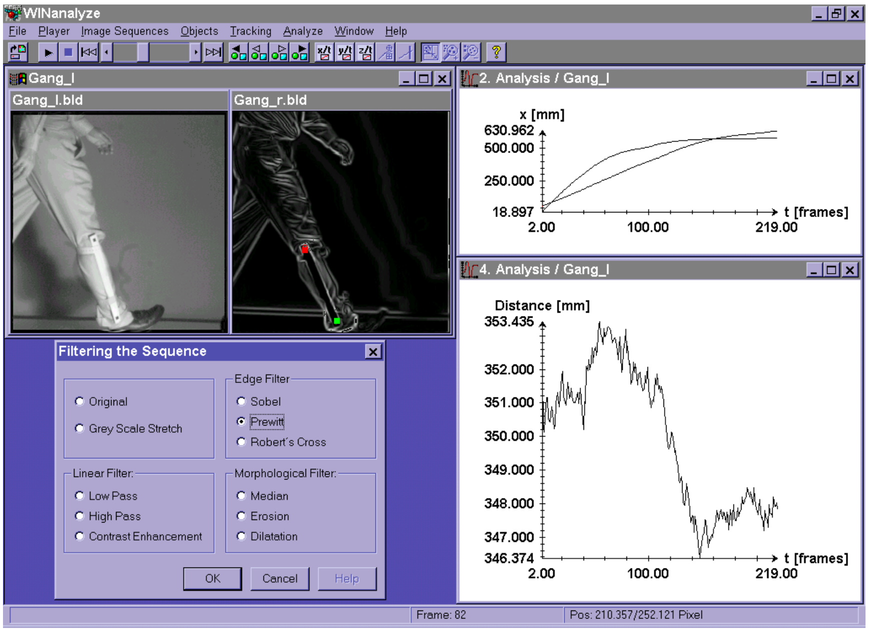
- Any arbitrary number of cameras can be used, as each is calibrated separately.
- Different types of cameras can be used; the user is not restricted to a special device.
- By allowing up to 200 calibration points, high precision calibration results can be achieved.

15.2.5 System integration—the software WINanalyze

The overall integration of the described algorithms and methods have led to the 32-Bit Windows software named WINanalyze. The design is object-oriented: Each single tracking object has its own adjustable parameters for the tracking algorithm, the template parameters and the visualization. The amount of object points for tracking is virtually unlimited (depending on the amount of memory of the PC system).

As input, WINanalyze reads in digital video files—either in binary raw data, or in the widely spread AVI format. By specifying several video files taken from several cameras, 3-D motion analysis can be achieved with up to 10 different camera views.

Several template matching algorithms are integrated which, in combination with customizable prefiltering of the image sequences, offer the user flexibility in tracking different objects under different conditions (for example, one tracking algorithm eliminates lighting vari-



Original
Grey Scale Stretch

Edge Filter
Sobel
[✓ Prewitt]
Robert´s Cross

Linear Filter:
Low Pass
High Pass
Contrast Enhancement

Morphological Filter:
Median
Erosion
Dilatation

Frame: 82
Pos: 210.357/252.121 Pixel

Figure 15.8: *WINanalyze—realization under Windows 95/NT. A stereo image sequence for gait analysis is shown, partially filtered with a high-pass filter. Two points were tracked on the leg. The chart on the upper right shows the horizontal trajectory of both points. In the lower right, the distance of both points along time is shown.*

ations, while another one takes color information into account (see Fig. 15.8).

15.3 Sample applications

15.3.1 Automotive industry

In car crash tests, markers are tracked on dummies and steering wheels. Car crash tests are nowadays typically recorded with digital high-speed cameras at 1000 frames/s with a resolution of 512×512 pixel. One second of recording, therefore, requires 256 MB of storage space. Tracking an object with WINanalyze throughout such a large sequence takes about 5 min on a state-of-the art PC. After the acquisition of the trajectories, the acceleration data is especially useful for analysis of the test. Comparisons of our motion tracking system with other acceleration measuring instruments showed a good correlation [9] (see Fig. 15.9).

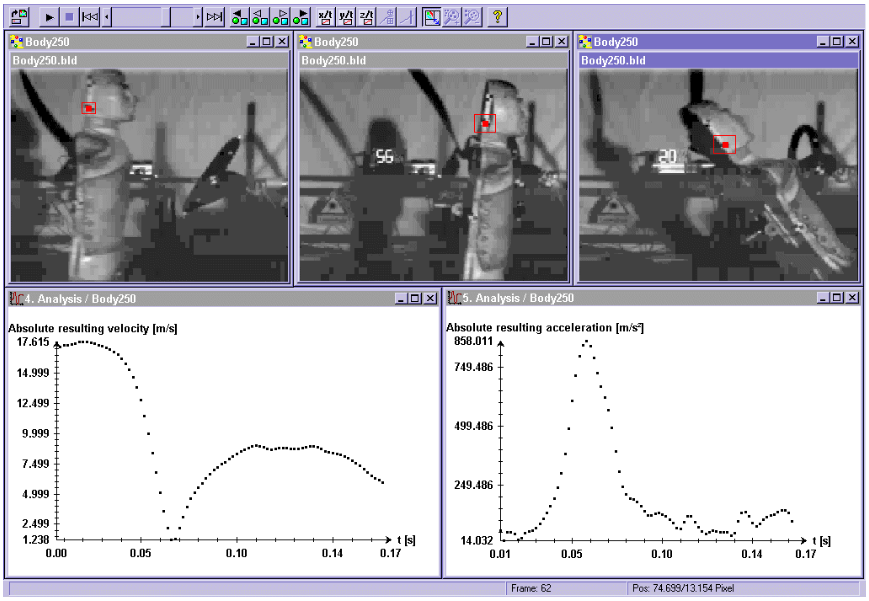


Figure 15.9: *Impact of a test dummy on a steering wheel in a car crash test experiment. Three images of the overall sequence are shown on the top. On the lower left, the first derivative diagram shows the strong decrease of the head's velocity at the time of the impact. At the lower right, the second derivative clearly shows the acceleration along time.*

15.3.2 Sports industry

To test the stability of sport shoes, ADIDAS [10] used a high-speed camera at 200 Hz to record the impact of the foot as seen from behind. The angle between the heel and the ground is calculated to compare the stability of different shoes. One image sequence consists of approximately 50 frames, in which four points have to be tracked (two on the ground for the horizontal axis, and two on the shoe for the heel axis). Within three days, 1000 of these sequences have to be analyzed. ADIDAS makes these test-rows approximately six times a year. Therefore, rapid analysis is vital for this kind of industrial application. With WINanalyze, the desired results are available after about 30 s (including tracking and angle calculation) on a PC Pentium 200 MHz. Figure 15.10 shows a sample out of a test sequence. Sometimes markers are applied to the shoes, sometimes features on the shoe and the ground are used for object tracking. Because of the strong deformations at the touchdown phase, template adaptation is very important for successful tracking in this application.

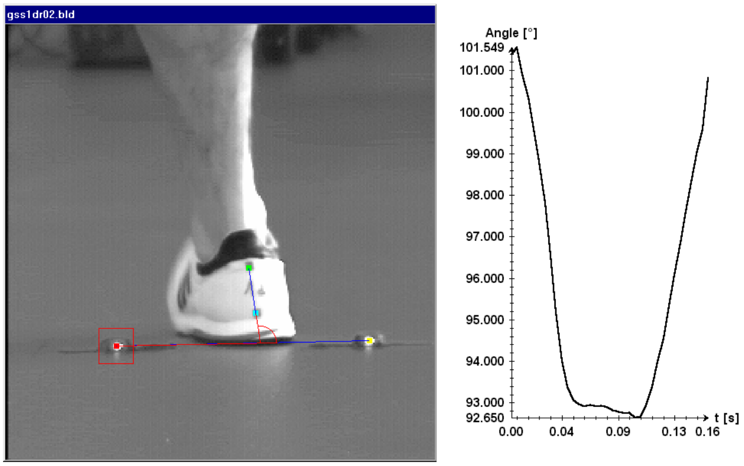


***Figure 15.10:** Shoe stability test example. The left image shows one frame of the sequence; two points on the shoe and two objects on the ground define the angle to be measured. The right image shows the calculated angle over time.*

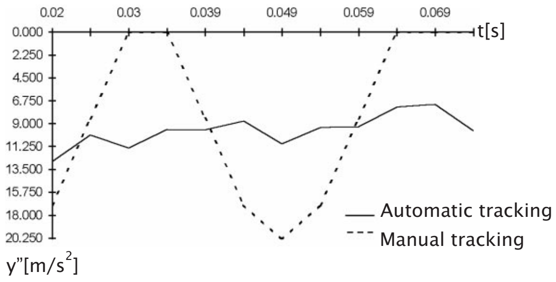


***Figure 15.11:** y''-t diagram.*

15.3.3 Physics

Often the computation of physical constants is used to verify the results of motion tracking. Therefore, an experiment to calculate gravity is presented here. A metal ring was released along a metal stick (used as a calibration model). The falling object was recorded with a high-speed camera at 202.7 frames/s with a resolution of 128×128 pixel. The sequence consisted of 20 frames.

With the correlation algorithm, the falling ring was tracked automatically. Additionally, the sequence was analyzed by a human operator with pixel accuracy. The vertical components of the trajectories were then differentiated and compared. Figure 15.11 shows the comparison of the second derivative of each automatically and manually tracked trajectory.

As can be clearly seen, the acceleration values of the manual trajectory are much less close to the correct value. To determine the earth gravity constant $g \approx 9.81\,\mathrm{m/s^2}$, both trajectories were approximated by a quadratic polynomial[1].

The now constant acceleration is calculated as $9.39\,\mathrm{m/s^2}$ with manual tracking, and $9.74\,\mathrm{m/s^2}$ with automatic tracking. The better correspondence to g is the result of the subpixel-precise object location used in our system.

This example summarizes the main advantages of the automatic tracking system over manual tracking: The acquisition of points is much faster and more precise. In addition, as long as there are distinct features on the object to be tracked, the appliance of markers is not even necessary.

15.4 Conclusion and remarks

Motion analysis when done manually is both ineffective and inaccurate. More efficient approaches for its automation make use of online (active) marker systems and marker-based off-line analysis tools feasible. The latest innovations use pattern recognition algorithms for more flexible motion tracking.

In the WINanalyze system presented here, object detection was automated by use of adaptive template matching. By applying subpixel precision methods, the matching accuracy could be enhanced by an average factor of 20. For one of the major problems in object tracking, the occlusion of objects, a new approach using neural nets is described. To gain real-world coordinates of the tracked pixel positions, a flexible camera calibration routine was added.

The system is used in many fields:

- In the industrial field for machine inspection and car crash tests, in combination with digital high-speed cameras [4].
- In biomechanics in combination with EMG devices for the analysis of human movement, such as clinical gait analysis [12].
- In sports for material research and athlete training analysis.

The system tracks well when the object to be located has some distinct features, and shows limited distortions from one frame to the next. However, under circumstances where the object of interest is not clearly visible, or can not be distinguished from its surroundings, improvements in object recognition have to be made. More *a priori* information about the scene and the properties of the objects in the

[1]The parabolic approximation of measurements in this experiment is a typical procedure in experimental physics (e.g., see [11]).

scenes have to be taken into account. Many researchers work on the modeling and recognition of objects for motion analysis. Therefore, the move from "low-level" to "model-based" image processing is the next step in enhancing automatic motion analysis.

15.5 References

[1] Childers, D. G., (1977). Laryngeal pathology detection. In *CRC Critical Reviews in Biomedical Engineering*, pp. 375–425.

[2] Tanabe, M., Kitajima, K., Gould, W. J., and Lambiase, A., (1975). Analysis of high-speed motion pictures of the vocal folds. *Folia Phoniatrica et Logopaedica*, **27**:77–87.

[3] Barnea, D. and Silverman, F., (1972). A class of algorithms for fast digital image registration. *IEEE Trans. Computers*, **C-21**:179–186.

[4] Bauer, N. and Frischholz, R. W., (1995). 3-D automatic motion analysis. In *ISATA 28th Symposium, Proceedings on Robotics, Motion and Machine Vision in the Automotive Industries, Stuttgart*. Craydon, England: Automotive Automation Limited.

[5] Frischholz, R. W. and Spinnler, K. P., (1993). A class of algorithms for real-time subpixel registration. In *Europto Series, Proceedings, Munich*, Vol. 1989.

[6] Seitz, P. and M., R. J., (1989). Optische Überauflösung mit CCD Kameras und digitaler Signalverarbeitung. In *Optical 3-D Measurement Techniques*. Karlsruhe: Herbert Wichmann Verlag.

[7] Frischholz, R., (1998). *Beiträge zur automatischen dreidimensionalen Bewegungsanalyse*. Dissertation, Universität Erlangen, Aachen: Shaker Verlag.

[8] Tsai, R. Y., (1986). An efficient and accurate camera calibration technique for 3-D machine vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Proceeding*, pp. 364–374.

[9] Schumann, H., (1995). *Durchführung von Kollisionsversuchen mit PKW, Dokumentation, Ausführung und mathematische Simulation*. Diploma thesis, Institut für Maschinenkonstruktionslehre und Kraftfahrzeugbau, Universität Karlsruhe.

[10] Thompson, M., Luethi, S., Kaelin, X., and Nigg, S., (1997). Effects of running shoe sole construction on dynamic foot stability. In *Proceedings 3rd Symposium on Footwear Biomechanics, Tokyo*. Tokyo, Japan: Metropolitan University.

[11] Feuerlein, R., Näpfel, H., and Schedl, E., (1993). *Physik, Mechanik*. München: BSV Verlag.

[12] Frischholz, R., (1997). Software zur automatischen Bewegungsanalyse. Biomechanik und Motorik, DVS Band 75. Hamburg: Czwalina Verlag.

16 Three-dimensional Image Metrology for Industrial Quality Control and Production

Horst A. Beyer

Imetric SA, Technopole, Porrentruy, Switzerland

16.1 Introduction

Three-dimensional image *metrology* has become a well-accepted technology for industrial quality control. Its use is spreading into production areas, either as an in-process quality control system or as a system controlling, in the sense of guiding, production machines. Three-dimensional image metrology is used in this contribution to refer to high-accuracy 3-D image metrology using digital images and targets. It is characterized by a relative accuracy of 1 part in 10,000 and better, where the relative accuracy is defined as the accuracy of the 3-D spatial coordinates of a point divided by the largest extent of the object.

The development of 3-D image metrology can be divided into the following phases (see Gruen [1] for an extensive overview). Early developments included the calibration of vidicon tubes for measurement applications [2, 3, 4] and position-sensitive devices [5, 6]. The first investigations with area array cameras were performed by El-Hakim [7], Real [8], and Haggrén [9]. In the mid-1980s, a small number of groups started to work on 3-D image metrology systems (e. g., El-Hakim [10], Haggrén

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

ISBN 0–12–379773-X/$30.00

[11], Real and Fujimoto [12], Gruen and Beyer [13]). The author of this contribution was fortunate to be part of one of these groups, working to develop a series of prototype quality control systems in 1985. By 1986, the typical accuracy achieved with *area array sensors*, usually *charge coupled devices* (CCDs), were in the order of 0.1 of the pixel spacing. In the following years, greatly improved target localization methods and calibration techniques were developed. These developments were building on existing techniques that were developed between the 1960s and the 1980s. Other advances included improvements in the camera and frame grabber electronics (e.g., [14, 15]). By 1992, the measurement accuracy with area array sensors approached 0.01 pixel in 3-D measurements (e.g., [16]).

In 1992, CCD sensors attained resolutions of 1000 × 1000 pixels, and the first portable camera with digital storage, the Kodak DCS100, became available. This type of camera was a breakthrough and still dominates the market. The Kodak DCS100 had a sensor with 1528 × 1024 pixels with a separate hard disk unit for image storage. The introduction of the Kodak DCS200 camera brought the next important step. It had an integrated disk which allowed 50 images to be stored. Once the disk was full, the camera was connected to a computer via a SCSI cable for image transfer. This camera promised to achieve a relative accuracy sufficient for many industrial applications. This led to a revolution in the use of metrology systems for industrial quality control. The new CCD-camera-based 3-D image metrology systems were so fast that users of this new technology were already analyzing the measurement results, when they were used to developing the film in a laboratory as required by traditional, film-based, measurement systems.

The first industrial 3-D image metrology systems were offered by several companies and installed in the early 1990s. The following years brought about large improvements in terms of turnaround speed, user friendliness, and accuracy. These were achieved through major improvements in algorithms and hardware. The high performance and specific know-how included in these systems has generated a niche market with a few highly specialized manufacturers. Most of the systems can be characterized by the following elements:

- Special targets (retroreflective targets, LEDs, high contrast object features, etc.) are used to signalize the points of interest. This is done to provide for an optimum contrast.
- These targets are either used as stick-on targets, as adapters to fit into tooling holes or other elements, or on touch probes.
- Special and/or adapted cameras are used where the geometry of the camera is more stable than in standard cameras.

- The relative accuracy (standard deviation of object coordinates divided by object size) of 3-D measurements is better than 1 part in 10,000.

The major factors that have led to a wider acceptance of 3-D image metrology systems in the industrial market are generally not as apparent. The most dramatic influence was the consistent use of so-called “coded” targets and the automatic establishment of the correspondence of standard targets. As late as 1994, users of 3-D image metrology systems had to manually identify 4 points in each image, and then have the system compute the orientation of the image. This was a very time consuming and tiring process. The use of “coded” targets allowed the user to simply press one button and have the system perform this task, with a much smaller error rate. This process was improved so much that by pressing one button a complete set of images can be measured automatically. In 1995 the labeling of standard targets, that is, targets without a code, still required manually identifying and labeling a point in at least two images. Today, the operator needs to label points in one image only or simply leave the (arbitrary) labeling completely to the software. These improvements reduced the time spend on one image from minutes to seconds. Other improvements, such as the integration of processing units into the camera and the increase in resolution, are equally tremendous achievements but did not contribute to the same extent to the reduction in measurement time and/or functionality/performance of the systems. The largest contribution to the acceptance of the systems in the marketplace is the fact that they have become sufficiently easy to use. Even production personnel uses them “just like a drill or a hammer.” In the next section, a few applications will show the challenges to be met by such systems, the material and methods used, and the implications and advantages for the user of such systems.

16.2 Geometry check of wing roots

The interface surfaces between wing and fuselage are areas where large forces must be transferred. It is thus very important that these interfaces conform to the defined geometry. Deviations lead, among other effects, to increased stress in these areas, and, thus, a lower life expectancy of the wing. The interface areas are checked before each wing is shipped to the integration site. In these checks the positions and angles of a large number of surfaces need to be verified.

Figure 16.1 shows a wing root with an operator posing with a camera (images are not taken that close). The camera has the typical circular flash to illuminate the targets. Different target adapters can be seen on

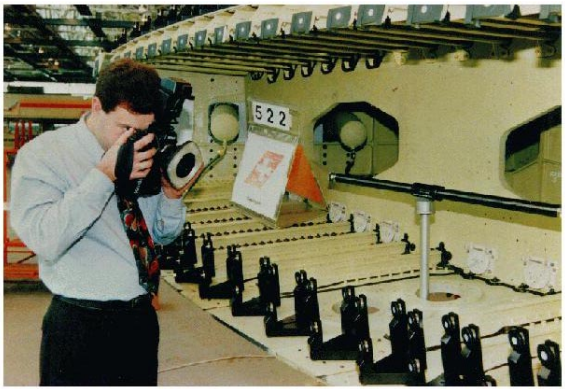


Figure 16.1: *Operator and wing root showing camera with flash, target adapters on wing and scale bar to the right (Photography courtesy of British Aerospace Airbus).*

the bottom and on the top of the wing root. On the right-hand side, a scale bar is visible.

Traditional techniques, that is, industrial theodolite systems, required that each wing remain completely stable for approximately 5 h. This in effect means that production is stopped for this period of testing, directly affecting throughput. The major aim of using a 3-D image metrology system was to reduce production interruption to a minimum. The solution to this occurred as a result of the development of special adapters (mechanical constructions with targets where the geometric relation of targets to mechanical references is precisely known, see Fig. 16.1) which can be placed and removed while production continues unaffected. This effectively reduced the time during which production must be stopped to the time required for acquiring the images, which is only 5 to 10 min. The complete process today includes the following steps:

- Application of the adapters and other targets. The exact placement of the adapters is not critical. Some of the other targets are placed in reference holes, which define the coordinate system of the wing.
- Two or more scale bars are placed within the measurement volume. They assure the traceability of the measurements to a national or an international standard.
- Images are taken (typically 20 to 30 images). An Imetric/Kodak DCS420 camera is used for this purpose. The images are stored on PCMCIA disks.
- The data is transferred from the PCMCIA disk to a desktop computer.

- The images are processed automatically using “coded” targets and approximations for the 3-D location of all other targets. Once all targets are measured in all images, the 3-D coordinates of all targets are computed, the coordinates are scaled using the certified values of the scale bars, and transformed into the coordinate system of the wing. The measurement results are then compared to the nominal values, and protocols are produced.

The total process takes only a few hours. This allows production to make sure that eventual deficiencies can be corrected in the jig before the next wing is build.

Another revolution involved handing over each system to production personnel. It was deemed more productive if the same personnel who build the wings also inspect them. The metrology department of that manufacturer prepared a detailed guide for this measurement task, trained the production personnel, and successfully handed over the complete job. This proved that a relatively complex technology such as high-accuracy 3-D image metrology can be successfully used by non-specialists with proper preparation and training. Two or more wings are inspected per day and further increases are expected to occur within the next years.

Due to the importance of accuracy, the requirements of this application are rather stringent. Any erroneous measurement resulting in a stringer being out of tolerance, could lead to large payments. On the other hand, a deficiency which is not detected may result in fit-up not working correctly. This could lead to production interruptions that are even costlier. The customer thus went through extensive acceptance and introduction procedures lasting for almost one year. One of the tests was to show the repeatability of the system. In this test, a repeatability of 0.015 mm was demonstrated in all three coordinate axes on a wing root spanning $2.8 \times 1.0 \times 0.5$ m, resulting in a repeatability of 1 part in 168,000. This is an outstanding result considering that a sensor with 1548×1024 pixels was used. The typical measurement accuracy in the images for this application approaches 0.01 pixel under practical conditions. This is only possible by adjusting every detail of the process.

16.3 Three-dimensional image metrology in shipbuilding

A shipyard that started to test the technology in 1993 and implemented their first system in 1994, is used here to show the impact 3-D image metrology had on shipbuilding (for more details, see [17]). The shipyard has used the systems for a large number of applications. Over the years they have implemented all improvements as they became available such that today they use the highest resolution cameras and latest software available.

With most production techniques, a ship is built by blocks called units, which can be economically manufactured in factories. The assembly of these blocks is usually achieved by leaving excess steel at the interfaces, which is cut during the erection process. One of the traditional techniques to determine the excess amount of steel was to position the unit on the ship and trace the line where the cut must occur. The unit is then removed from the ship and the excess steel is cut away. The unit is then positioned on the ship and welded to the existing structure. This process is extremely time consuming as it involves three moves of the unit versus one, and it is not accurate. Using 3-D image metrology, the interfaces are measured while the unit is sitting in the factory or in an intermediate storage area. The interface area on the ship is also measured. Comparing the two measurements allows the exact cut line on the unit to be determined. As this user of the technology was an early adopter, it provides a good example to discuss the effect of the improvements. The typical measurement process consists of the following steps: the locations of the targets on the interface area are defined in CAD. A map with all target locations is provided to the persons applying the targets. Simple magnetic targets with a retro-reflective surface are used as the accuracy requirements are not too extreme. This task is relatively easy and requires only 1 to 2 h for the 5 sides of a unit (see Fig. 16.2). Images are acquired with the help of a cherry picker. This typically takes about 1 h when all 5 sides of a unit are to be measured. The images are thereafter transferred to the analysis system. There, the orientations are computed automatically using the coded targets. The labeling of all other targets can be done automatically by transforming the data into the coordinate system of the ship and using the nominal locations of the targets. Another option is to use the coordinates from a former measurement of a unit or to label them manually. Using the currently available software, the processing can be achieved within 1 h.

The initial system used Kodak DCS200 cameras. Whenever the disk of the camera was filled, that is, after 50 images, the camera had to be brought to the office to download the data. This was a significant drawback for larger projects. Furthermore, coded targets and other automated algorithms were not available at that time. Tests in 1997 showed that the analysis time could be improved by almost an order of magnitude due to the introduction of those technologies. The use of higher resolution cameras and the storage on PCMCIA disks allowed bigger projects to be tackled, that is, projects with a larger number of images and larger objects (up to the size of a ship).

An additional benefit of the introduction of the system was that a thorough statistical process control could be introduced, which al-

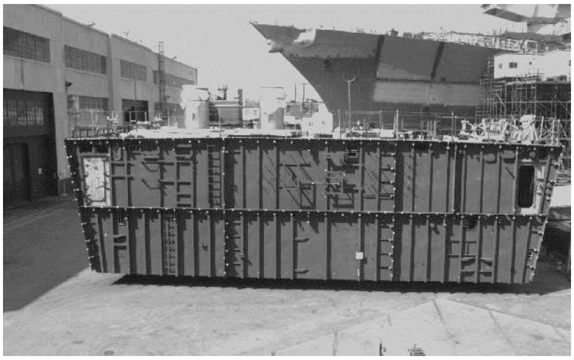


***Figure 16.2:** Typical unit with targets applied to section. All five faces of this unit are measured in one operation. The background shows the units on the assembled ship (Photography courtesy of Bath Iron Works).*

lows, for example, the quality of each unit to be tracked from ship to ship. An unexpected but major benefit was that work-related accidents were eliminated. A major achievement is, furthermore, that the personnel who used to run transits and other measurement systems could be trained to use this high-technology equipment. One could say that this is a typical example of what proper motivation and training can do. In terms of technology and accuracy, this shipyard has progressed throughout the complete range of technology. The accuracy requirements per se do not sound overly stringent with a 1 mm accuracy over a unit. It must be considered, though, that the environmental conditions are very harsh. In summer, the units are very hot and in winter, measurements are performed at temperatures significantly lower than -20 °C. Thus, the relative accuracy of 1 part in 30,000 to 50,000 must be considered as very good.

16.4 Machine control and TI^2 technology

The majority of applications of 3-D image metrology systems are to verify whether or not an object, for example, a tool or a part, conforms to its nominal geometric shape. Although the systems are very useful, they still only allow assessing whether or not something is out of tolerance and, thus, subject to rework and, in the worst case, discarding of the part. The feedback is indirect, as the fabrication is not directly controlled. Closing this loop allows controlling a machine with the 3-D image metrology system while the part is manufactured, thereby effectively improving the accuracy and reducing errors.

This is exactly where the next revolution in the use of 3-D image metrology systems in industrial applications is headed. This changes the role from that of an “inspection system” to an integral part of a production system, that is, a “machine control” system. One approach to *machine control* was developed by a group of companies and is called TI^2 technology. The name TI^2 stems from the original products used in the initial product development, that is, the Tricept robot from Neos Robotics (Sweden), the Imetric 3-D image metrology system from Imetric (Switzerland), and the IGRIP simulation software from Deneb (U.S.).

The TI^2-system technology is based on the idea of controlling the location of an end effector directly in relation to the part by using 3-D image metrology instead of expensive mechanical systems.

The concept of the TI^2-system technology consists of the following: the tool of the CNC-machine, a robot or another system, is equipped with targets and precalibrated such that the relation between the targets and the *tool center point* (TCP) location and orientation is precisely known. The part or object is prepared for TI^2 by applying targets to it. The targets need not be applied to precise positions, only some reference locations must be known or a CAD model of the part must be available. A 3-D image metrology system is used to determine the precise 3-D coordinates of the part in the coordinate system defined by reference locations and/or the CAD model. Now, the part can be placed in front of the TI^2 system. The TI^2 system then uses the targets on the tool and the targets on the part to determine the 6 degrees of freedom relation between the part and the NC-machine/robot. This is a major improvement over the use of mechanical indexing schemes. Once the 6 degrees of freedom are determined, the NC-program is updated to reflect the location of the part in the coordinate system of the NC-machine/robot and the program is executed.

Figure 16.3 shows one setup of the system for the machining of an attachment hole of an airplane tailcone. In this application, the coordinate system is determined via the use of a CAD model of the exterior of the tailcone. A 3-D image metrology system is used to compute the 3-D coordinates and a special software package is used to perform the best fit between the points and the CAD surface. The piece is then simply placed in front of the NC-machine, in this case a very rigid robot. The system determines the relation between the piece and the NC-machine, updates the program, and machines the hole. This application demonstrates several features of the technology. First, there is no (expensive) mechanical positioning system required. Second, there is no need to build a custom tool to hold the piece and to position the tool to machine the hole. Third, the system can verify its work while it is being performed.

The advantages of the TI^2 technology concept are very well demonstrated with this application. It required approximately two days to

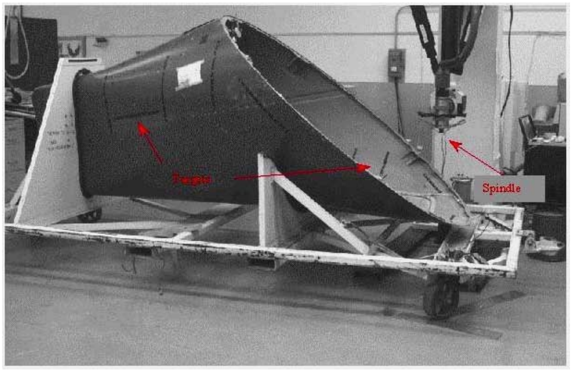


Figure 16.3: *Tailcone of an airplane with targets on the outside and inside. On the right side, the spindle with target adapters can be seen (Image courtesy of Boeing).*

develop the complete application. In comparison to this, it would probably take several weeks to design the fixtures and tools, have them built and certified, and perform the same machining with probably less accuracy. The system does not require an expensive fixture as needed in the case of a traditional mechanical approach. In the mechanical approach, the part would have to be indexed via some references with respect to the holding fixture. With TI^2, the indexing is directly done with the 3-D image metrology system and the targets on the part, providing higher accuracy and lower cost.

Figure 16.4 shows the bulkhead of an airplane. There are approximately 4000 holes to be drilled in this part. The TI^2 system can perform this over 4 times faster and more accurately than current technologies. The process starts again by applying targets to the bulkhead parts and the tool. Then, the 3-D coordinates are determined with a standard system. The coordinate system is defined via reference holes on the structure holding the piece, and verified using the CAD model of the piece. To start machining, the TI^2 system is approximately positioned in front of the bulkhead, and the system is told to start the processing. Basically, the system can automatically determine which part and which area of the part it is supposed to machine. If this automatic determination is not possible, for example, because the piece was placed too far away or would destroy the part in case it moves, it will inform the operator. If these checks are completed successfully, the system will start machining. In this application demonstration, the system drills holes and performs some routing. The geometric performance of the system is demonstrated in the following way. The system is removed and placed in a different position in front of the part. The piece is also rotated by approximately 360° around its axis to a different position.

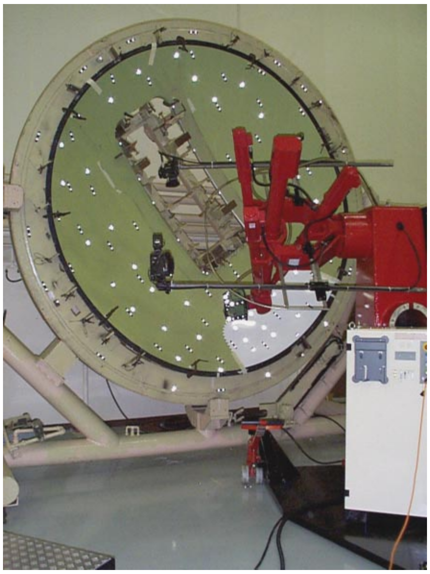


Figure 16.4: *Bulkhead of an airplane with TI^2 system (Image courtesy of Boeing).*

The system is capable of drilling identical holes without even touching the metal. This proves the outstanding accuracy of the TI^2 system, which far surpasses conventional techniques.

There are many applications to which this technology can be applied. Basically, any larger object with a complex geometry is an excellent application. These applications are found in aerospace, automotive, shipbuilding, facilities, etc. The advantage of indexing with 3-D image metrology is tremendous.

16.5 Developments

The integration of 3-D image metrology into industrial production machinery, such as big CNC-machines, imposes new requirements that are very difficult to meet. The systems must be extremely robust from an algorithmic point of view as there is no operator to intervene. The results must be immediately available, that is, within 1 s for cameras with 3000 $\times$ 2000 pixel sensors. The complete 3-D image metrology system must be capable of operating without interruption for many years. Parts, that is, cameras that fail, must be replaceable within minutes, often without being able to shut down the machinery. Cameras routinely used in 3-D image metrology systems would not survive long enough in this environment and are not capable of reliably meeting

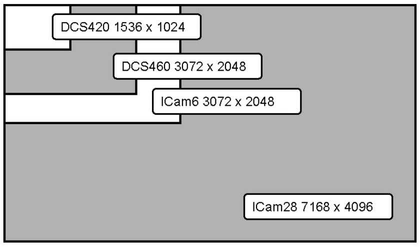


Figure 16.5: *Relative size of some sensors used in various 3-D image metrology systems.*

these performance requirements. New cameras are being developed to meet these performance requirements. Some are based on standard components that are adapted as much as possible to the rough environment and have integrated processing units. Others are specifically for 3-D image metrology and the rigors of a production/machining environment. The latter instance varies radically from the traditional approach, where cameras that were originally designed for some other imaging application were “misused” for 3-D image metrology. Achieving this goal requires the redesign of most components including flash, optics, CCD sensor, analog electronics, processing unit, software, and housing. The *metrology camera system* developed by Imetric, called ICam for Imetric Camera, will be used to highlight some of the features of such cameras.

The ICam series of cameras provides resolutions from 1500×1000, over 3000×2000, to 7000×4000 pixels. It is based on CCD sensors manufactured by Philips. Figure 16.5 shows the relative size of some of the currently used CCD sensors and the sensors used in the ICam. The difference in size comes from the different sensor element spacing, which is 12 μm for the Philips sensors versus the 9 μm spacing of the Kodak sensors used in the Kodak DCS420 and DCS460 cameras. Both cameras provide a dynamic range exceeding the typically used 8-bit per pixel (see Chapter 7). To take advantage of this capability a 12-bit analog-to-digital converter is used and the imagery is stored as 16-bit data, thus doubling the amount of data to be handled.

Typically standard optics, specially selected for 3-D image metrology, were used in all systems. While these optics are certainly excellent for standard photography, 3-D image metrology has particular requirements. Optimizing an optical system for solely this purpose promises better performance.

The very high throughput rate of these cameras required careful selection of the CCD sensor and special design of the analog electronics of the sensor. Typical sensors provide a data rate of about 10 MPixel/s, thus taking just a little less than 1 s to read out an image. The ICam

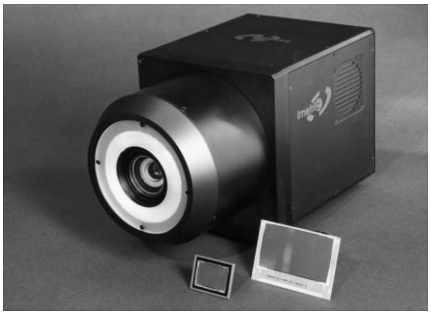


Figure 16.6: *ICam6 with 3000 × 2000 and 7000 × 4000 CCD sensors from Philips (Courtesy of Imetric SA).*

was designed to have a data rate of 32 Mpixel/s, delivering 5 images/s from a 3000 × 2000 CCD sensor. This allows significantly more time for processing but requires a radically new design of the readout, storage, and transfer. With the 7000 × 4000 pixel sensor delivering 56 Mbytes per image it became obvious that the data should be processed inside the camera. The resulting data is only a few Kbytes, resulting in significant data reduction. This has the added advantage that the processing time in multicamera arrangements, such as those used in TI^2 and other machining applications, is independent of the number of cameras used.

Making cameras self-contained makes maintenance and replacement of cameras much simpler. The ICam can basically be removed and replaced with another camera without needing to interrupt the system. The camera contains all the calibration information required for the 3-D image metrology system. The housing and all internal workings had to be designed to survive the rigors of machining environments with strong vibrations, dirt, and dust. The housing totally encapsulates the camera and protects it from the environment, even in graphite machining environments, which are extremely hostile to electronic components.

Figure 16.6 shows an ICam6 with the 3000 × 2000 and 7000 × 4000 pixel sensors from Philips used in the ICam6 and ICam28 cameras, respectively. It shows the integrated flash and lens system. The box-like part of the camera houses the processing unit. The camera uses pressure ventilation to keep the temperature close to that of the environment with provisions to cool as well as to protect it and the optics from the environment.

The top-level camera, the ICam 28 (28 for the number of pixels of the CCD sensor), will effectively double the accuracy that can be attained with this new camera as compared to the currently used cameras that are usually based on 3000 × 2000 sensors. This camera will have an accuracy comparable to those of the most precise metrology systems available on the market. It will, thus, open up a whole range of new applications for 3-D image metrology

16.6 Conclusions

The applications show that 3-D image metrology has developed to a mature technology, which is providing genuine return to the practical users of these systems. The importance of algorithmic advances and hardware improvements are well demonstrated. While one usually looks at the awesome improvement in computer hardware, one must state that in this particular instance the algorithmic improvements surpass those of raw computing power. Future algorithmic improvements will further enhance functionality by making them even easier to use, faster, and more robust.

16.7 References

[1] Gruen, A., (1996). Development of digital methodology and system. In *Close Range Photogrammetry and Machine Vision*, K. Atkinson, ed., pp. 78–99. Lathersnwhzel, Scotland, U.K.: Whittles Publishing.

[2] Kratky, V., (1979). Real-time photogrammetric support of dynamic three-dimensional control. *PE&RS*, **45**:1231–1242.

[3] Pinkney, H., (1987). Theory and development of an on-line 30 Hz video photogrammetry system for real-time 3-dimensional control. In *ISPRS Commission V Symposium, Stockholm. Presented paper.*

[4] Wong, K., (1969). Geometric distortions in television imageries. *PE*, **35(5)**: 493–500.

[5] Reece, D., (1981). *A SELSPOT-based data acquisition system for use in a clinical motion study laboratory.* Master thesis, Department of Electrical Engineering and Applied Physics, Case Western Reserve University.

[6] Woltring, J., (1975). Calibration and measurement in 3D-monitoring of human motion by optoelectronic means I. Preliminaries and theoretical aspects. *Biotelemetry*, **2**:169–196.

[7] El-Hakim, S., (1983). Photogrammetric robot-vision. In *Proc. ASP Fall Convention. Salt Lake City, September, 1983*, pp. 287–293. ASPRS.

[8] Real, R., (1983). Matrix camera with digital image processing in photogrammetric applications. In *Proc. ASP Convention, Washington D.C., March 1983*, pp. 255–266. ASPRS.

[9] Haggrén, H., (1984). New vistas for industrial photogrammetry. In *IAPRS, Commission V, ISPRS Congress Rio de Janeiro*, Vol. 25(A5), pp. 382–391. ISPRS.

[10] El-Hakim, S., (1986). A real-time system for object measurement with CCD cameras. *IAPRS*, **26(5)**:363–373.

[11] Haggrén, H., (1986). Real-time photogrammetry as used for machine vision applications. *IAPRS*, **26(5)**:374–382.

[12] Real, R. and Fujimoto, Y., (1986). Stero image transfer system with fast digital video processors and merged graphics display. In *Proc. ASP Convention, Washington D.C., March 1986*, pp. 272–283. ASPRS.

[13] Gruen, A. and Beyer, H., (1986). Real-time photogrammetry at the digital photogrammetric station (DIPS) of ETH Zurich. *Canadian Surveyor*, **41(2)**: 181–199.

[14] Bayer, B. E., (1976). Color imaging array, U.S. Patent No. 3,971,065.

[15] Raynor, J. and Seitz, P., (1990). The technology and practical problems of pixel-synchronous CCD data acquisition for optical metrology Applications. *SPIE*, **1395**:96–103.

[16] Beyer, H., (1992). *Geometric and radiometric analysis for a CCD-camera based photogrammetric close-range system*. PhD thesis No. ETH-9701, Federal Institute of Technology, Zurich, Switzerland.

[17] Johnson, G., (1994). Practical integration of vision metrology and CAD in shipbuilding. In *Proceedings of the ISPRS Congress, 1994, Commission V, Working Group V/3*.

17 Reverse Engineering Using Optical Range Sensors

Stefan Karbacher[1], Gerd Häusler[1], and Harald Schönfeld[2]

[1]Lehrstuhl für Optik, Universität Erlangen-Nürnberg, Erlangen, Germany
[2]Production Technology Systems GmbH, Fürth, Germany

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

17.1 Introduction

Optical 3-D sensors are used as tools for *reverse engineering* to digitize the surface of real 3-D objects (see Chapter 20). Common interactive *surface reconstruction* is used to convert the sensor point cloud data into a parametric CAD description (e. g., NURBS). We discuss an almost fully automatic method to generate a surface description based on a mesh of curved or flat triangles.

Multiple range images, taken with a calibrated optical 3-D sensor from different points of views, are necessary to capture the whole surface of an object and to reduce data loss due to reflexes and shadowing. The raw data is not directly suitable for import in CAD/CAM systems, as these range images consist of millions of single points and each image is given in the sensors coordinate system. The data usually are distorted, due to the imperfect measuring process, by outliers, noise and aliasing. To generate a valid surface description from this kind of data, three problems need to be solved: The transformation of the single range images into one common coordinate system (*registration*), the *surface reconstruction* from the point cloud data to regain object topology and the manipulation of the surface geometry to eliminate errors introduced by the measurement process and to reduce the amount of data (*surface modeling* and *smoothing*). Commonly used software directly fits *tensor product surfaces* to the point cloud data (see Volume 1, Fig. 20.29). This approach requires permanent interactive control by the user. For many applications, however, for example, the production of dental prostheses or the "*3-D copier*," this kind of technique is not necessary. In this case the generation of *meshes of triangles* as surface representation is adequate.

We work on building up a nearly automatic procedure covering the complex task from gathering data by an optical 3-D sensor to generating meshes of triangles. The whole surface of an object can be scanned by registering range images taken from arbitrary positions. Our procedure includes the following steps (see Fig. 17.1):

1. **Data acquisition:** Usually multiple range images of one object are taken to acquire the whole object surface. These images consist of range values arranged in a matrix, as on the camera's CCD chip.
2. **Calibration:** Measuring a standard with an exactly known shape, a polynomial for transforming the pixel coordinates into metrical coordinates is computed. The coordinate triplets of the calibrated range image have the same order as the original pixel matrix. This method calibrates each measurement individually. As a result each view has its own coordinate system.
3. **Surface registration:** The various views are transformed into a common coordinate system and are adjusted to each other. As the sen-

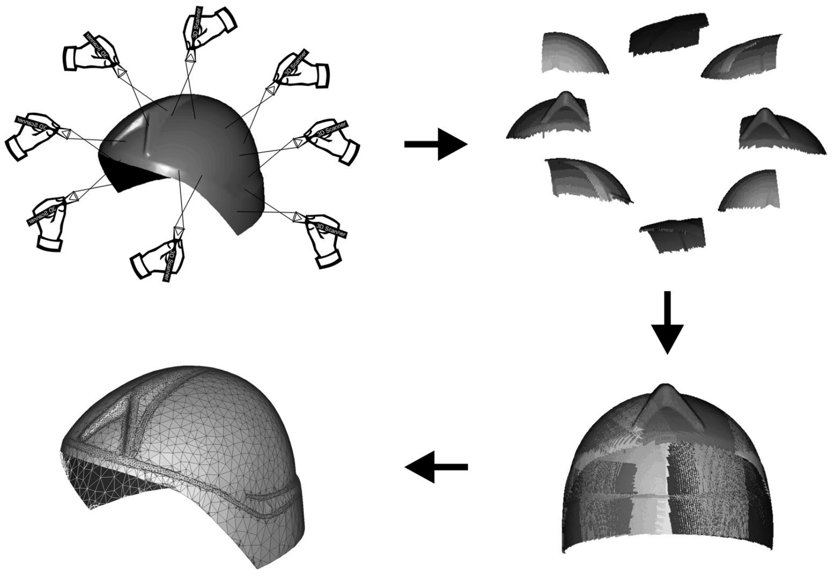


Figure 17.1: *Data acquisition, registration and surface reconstruction of a firefighter's helmet.*

sor positions of the different views are not known, the transformation parameters must be determined by an accurate localization method. First the surfaces are coarsely aligned one to another with a feature-based method. Then a fine-tuning algorithm minimizes the deviations between the surfaces (global optimization).

4. **Surface reconstruction:** The views are merged into a single object model. A surface description is generated using a mesh of curved triangles. The order of the original data is lost, resulting in scattered data.
5. **Surface modeling and smoothing:** A new modeling method for scattered data allows interpolation of curved surfaces only from the vertices and the surface normals of the mesh. Using curvature dependent mesh thinning, it provides a compact description of the curved triangle meshes. Measurement errors, such as *sensor noise*, *aliasing*, *calibration* and *registration errors*, can be eliminated without ruining the object edges.

In this chapter we give an overview on this reverse engineering method and show some results that were modeled with our software system SLIM$^{3\text{-D}}$. As the mathematical basis of the new approach for modeling of scattered 3-D data is not dealt with in the other chapters of this handbook, it will be discussed here.

17.2 Related work

17.2.1 Optical three-dimensional sensors

For detailed information in optical 3-D sensors for reverse engineering, refer to Volume 1, Chapters 18–20.

17.2.2 Calibration

There are two basic approaches using calibration methods for typical optical 3-D sensors. *Model-based calibration* tries to determine parameters of a sensor model that describe the imaging properties of the sensor as closely as possible. A few test measurements are needed to determine the coefficients for distortion and other aberrations. Imaging errors that are not covered by the sensor model, however, may impair calibration accuracy. This approach is discussed in Volume 1, Chapter 17.

Alternatively, an arbitrary *calibration function* (usually a polynomial), whose parameters are determined by a series of test measurements of a known standard object, can be used ([1, 2] and Volume 1, Section 20.4.2). The advantages of this approach are that a mathematical model of the sensor is not necessary and that the underlying algorithms are straightforward and robust in implementation. Drawbacks are the requirement of complex standards, which may limit the size of the field of view, and the fact that registration of multiple views is not implicitly solved during the calibration process.

17.2.3 Registration

The different views are usually aligned by pairs. First, the transformation between two neighboring views is roughly determined (*coarse registration*). In practice, this transformation is often found by manually selecting corresponding points in both views. Extraction of features, like edges or corners, combined with *Hough* methods, as in our algorithm, are used to compute a rough transformation more quickly [3]. Other methods try to determine the transformation parameters directly from the data, without explicit feature extraction. They are based on mean field theory or genetic algorithms or on curvature analysis (see [4, 5, 6]). An alternative approach is discussed in Volume 1, Section 20.5.

After finding an approximative transformation a *fine registration* procedure minimizes remaining deviations. The well-known *iterated closest point* (ICP) algorithm is used to find for every point in the first view the closest point of the surface in the second view (see [7]). The corresponding pairs of points are used to compute the transformation parameters. In every step of the iteration the correspondence is improved and finally leads to a minimum of the error function. Due to nonoverlapping areas of the views and noise, this minimum is not guaranteed to be the desired global minimum. Thus, additional processing of noncorresponding points is often done. Our algorithm combines the ICP with *simulated annealing* to avoid local minima.

17.2.4 Surface reconstruction and smoothing

The reconstruction of the topology of an object from a cloud of sampled data points can be solved by means of *graph theory* (see [8, 9]). At present, this approach is of little importance, as it is difficult to handle the amount of data provided by modern optical 3-D sensors. Furthermore, only the measured data points can be interpolated exactly; errors cannot be smoothed out.

Volumetric approaches have been most often used in recent years. These are based on well-established algorithms of *computer tomography* like *marching cubes* (see Volume 2, Section 28.3.2) and therefore are easily implemented. They produce approximated surfaces, so that error smoothing is carried out automatically. The method of Hoppe et al. [10, 11] is able to detect and model sharp object features but allows the processing of some 10,000 points only. Curless and Levoy [12] can handle millions of data points, but only matrix-like structured range images can be used. No mesh thinning is done, so a huge amount of data is produced.

The usage of *topology information* provided by the range images enables faster algorithms and more accurate results. For that reason, researchers have proposed several methods for merging multiple range images into a single *triangular mesh* (see [13, 14], and Section 20.5.3).

Such methods require special efforts for error smoothing. Our method includes an effective *smoothing filter* [15]. In contrast to other surface reconstruction methods it is able to smooth single images without significant loss of details. The other methods require redundant information. High-quality smoothing is possible only in overlapping areas of different images.

Filters for smoothing polyhedral meshes without usage of redundant information are still undergoing intense research. Lounsbery [16] uses a generalization of a *multiresolution analysis* based on *wavelets* for this purpose. Unfortunately, this approach works solely on triangular meshes with *subdivision connectivity*.[1] A filter that works on general meshes was proposed by Taubin [17]. He has generalized the *discrete Fourier transform*, in order to realize *low-pass filters*. However, the translation of concepts of *linear signal theory* is not the optimal

[1]All vertices (with singular exceptions) have the same number of neighbors.

choice. Surfaces of 3-D objects usually consist of segments with low bandwidth and transients with high frequency between them. They have no “reasonable” shape, as it is preconditioned for linear filters. “Optimal” filters like *Wiener* or *matched filters* usually minimize the *root mean square* (RMS) error. Oscillations of the signal are allowed if they are small. For visualization or milling of surfaces curvature variations are much more disturbing than small deviation from the ideal shape. A smoothing filter for geometric data should therefore minimize curvature variations and try to reinduce an error that is smaller than the original distortion of the data. These are the requirements we considered when we designed our new smoothing method.

17.3 Three-dimensional sensors

The right 3-D sensor to be used depends on the object to be digitized, that is, its size and surface properties. For details refer to Volume 1, Chapters 18–20.

17.4 Calibration

Optical sensors generate distorted coordinates $\boldsymbol{x}' = [x', y', z']^T$ because of perspective, aberrations and other effects. For realworld applications a calibration of the sensor that transforms the sensor raw data $\boldsymbol{x}'$ into the metrical, Euclidean coordinates $\boldsymbol{x} = [x, y, z]^T$ is necessary.

We present a new method for calibration of optical 3-D sensors [1]. An arbitrary polynomial is used as a *calibration function*. Its coefficients are determined by a series of measurements of a *calibration standard* with exactly known geometry. A set of flat surface patches is placed upon it. These intersect virtually at exactly known positions $\boldsymbol{x}_i$. After measuring the standard, an interpolating polynomial $\boldsymbol{p}$ with $\boldsymbol{x}_i = \boldsymbol{p}(\boldsymbol{x}'_i)$ can be found. Due to aberrations, the digitized surface patches are not flat. Therefore polynomial surfaces are approximated to find the virtual intersection points $\boldsymbol{x}'_i$. In order to fill the whole measuring volume with such calibration points, the standard is moved on a translation stage and measured in many positions.

The intersection points can be calculated with high accuracy, as the information of thousands of data points is averaged by surface approximation. As a result, the accuracy of the calibration method is not limited by the sensor noise. This method is usable for any kind of sensor and, in contrast to other methods, requires no mathematical model of the sensor and no localization of small features, like circles or crosses.

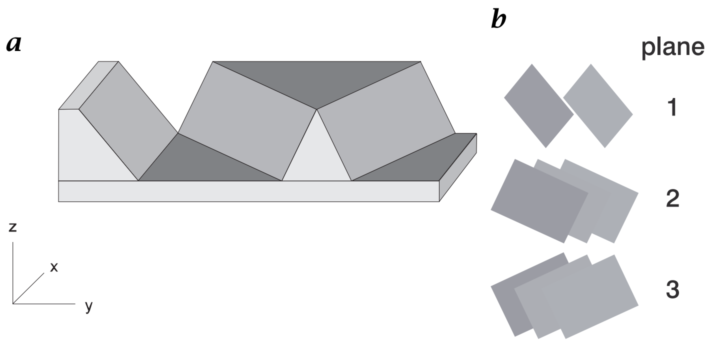


Figure 17.2: ***a*** *Calibration standard with three tilted planes; and* ***b*** *eight range images of these planes.*

17.4.1 Example

In the following example a phase-measuring sensor is calibrated using a block of aluminum with three tilted planes (Fig. 17.2), that is moved in y direction in small steps. After every step a picture is taken. About 52 range images of the 3 planes are generated. Eighteen images of the same plane define a class of parallel calibration surfaces.

Polynomials of order 5 are fit to the deformed images of the 3 planes. The polynomials of each class are intersected with polynomials of the other two classes (Fig. 17.3a). The points of intersection are spread throughout the whole field of view of the sensor (Fig. 17.3b). The position of the intersections in metrical coordinates $\boldsymbol{x}_i$ can be computed from the geometry of the standard and its actual translation. A polynomial $\boldsymbol{p} = [p_x(\boldsymbol{x}'), p_y(\boldsymbol{x}'), p_z(\boldsymbol{x}')]^T$ for transforming the measured intersection positions to the known positions is approximated by *polynomial regression*.

17.4.2 Results

Calibration of a range image with 512×540 points takes about 3 s on an Intel P166 CPU. Calibration error is less than 50 % of the measurement uncertainty of the sensor. It is sufficient to use a calibration polynomial of order 4, as coefficients of higher order are usually very close to zero.

17.5 Registration

Usually multiple range images of different views are taken. If the 3-D sensor is moved mechanically to defined positions, this information can be used to transform the images into a common coordinate system. Some applications require sensors that are placed manually in arbitrary

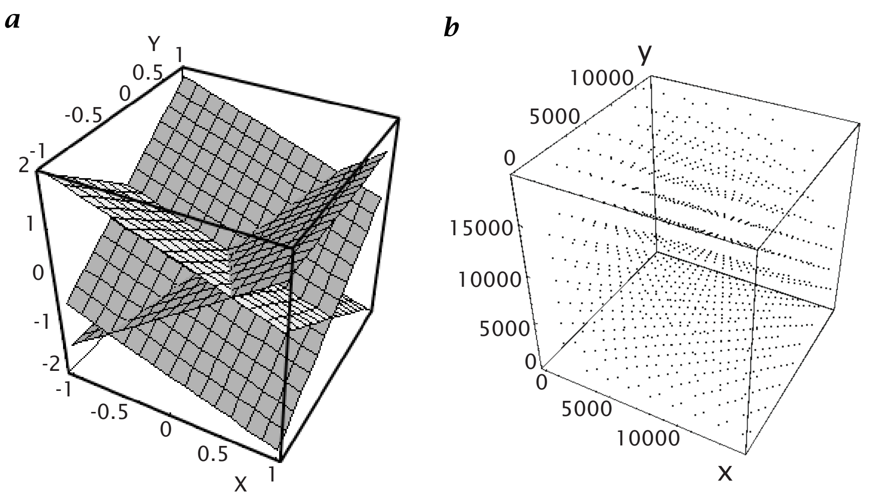


Figure 17.3: ***a*** *Intersection of three polynomial surfaces (one from each class) in arbitrary units; and* ***b*** *field of view with all measured intersection points in metric coordinates.*

positions, for example, if large objects like monuments are digitized. Sometimes the object has to be placed arbitrarily, for example, if the the top and the bottom of the same object are to be scanned. In these cases the transformation must be computed solely from the image data.

17.5.1 Coarse registration

We present a procedure for registration of multiple views that is based on *feature extraction*. It is independent of the sensor that was used. Thus it may be applied to small objects like teeth, and to large objects like busts.

Zero-dimensional *intrinsic features*, for example, corners are extracted from the range images (or congruent gray-scale images). The detected feature locations are used to calculate the translation and rotation parameters of one view in relation to a master image. Simultaneously, the unknown correlations between the located features in both views are determined by *Hough* methods. To allow an efficient use of Hough tables, the 6-D parameter space is separated into 2-D and 1-D subspaces (see Ritter [3]).

If intrinsic features are hard or impossible to detect (e. g., on the fireman's helmet, Figs. 17.1 and 17.13), artificial markers, which can be detected automatically or manually, are applied to the surface. The views are aligned to each other by pairs. Due to the limited accuracy of feature localization, deviations between the different views remain.

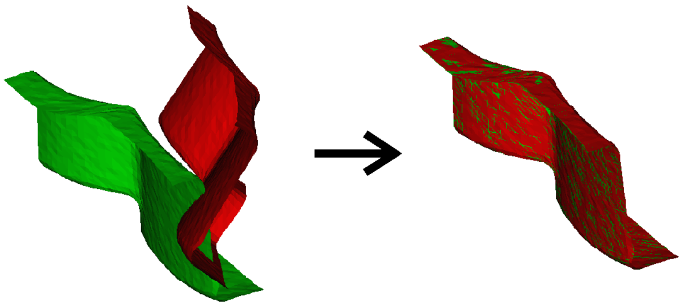


Figure 17.4: Fine registration of two noisy views with very high initial deviation ($\alpha = -30°$, $\beta = 50°$, $\gamma = 40°$, $x = 30$ mm, $y = 60$ mm, $z = -40$ mm).

17.5.2 Fine registration

A modified ICP algorithm is used for minimizing the remaining deviations between pairs of views. Error minimization is done by *simulated annealing*, so in contrast to classic ICP, local minima of the cost function may be overcome. As simulated annealing leads to a slow convergence, computation time tends to be rather high. First results with a combination of simulated annealing and *Levenberg-Marquardt*, however, show even smaller remaining errors in much shorter time: The registration of one pair of views takes about 15 s on an Intel P166 CPU. If the data is calibrated correctly, the accuracy is limited only by sensor noise.

Figure 17.4 shows the result of the registration for two views with 0.5 % noise. One was rotated by approximately 50°. The final error standard deviation was approximately σ_{noise} (standard deviation of the sensor noise).

17.5.3 Global registration

Using only registration by pairs, closed surfaces can not be registered satisfactorily. Due to accumulation of small remaining errors (caused by noise and miscalibration) a chink frequently develops between the surface of the first and last registered view. In such cases the error must be minimized globally over all views. One iteration fixes each view and minimizes the error of all overlapping views simultaneously. About 5 of these global optimization cycles are necessary to reach the minimum or at least an evenly distributed residual error. Global registration of an object that consist of 10 views takes approximately 1 h. If n surfaces overlap at a certain location, the global registration can reduce the final registration error at this location down to $\sigma_{\text{noise}}/\sqrt{n}$.

17.6 Surface reconstruction

We now present a new method to reconstruct the object surface from multiple registered range images. These consist of a matrix of coordinate triples $\boldsymbol{x} = [x, y, z]^T_{n,m}$. The object surface may be sampled incompletely and the sampling density may vary, but should be as high as possible. Beyond that, the object may have arbitrary shape, and the field of view may even contain several objects. The following steps are performed to turn this data into a single mesh of curved or flat triangles:

Mesh generation. Because of the matrix-like structure of the range images, it is easy to turn them into *triangular meshes* with the data points as vertices. For each vertex the surface normals are calculated from the normals of the surrounding triangles.

First smoothing. In order to utilize as much of the sampled information as possible, smoothing of measuring errors like noise and aliasing is done before mesh thinning.

First mesh thinning. Merging dense meshes usually requires too much memory, so mesh reduction often must be carried out in advance. The permitted approximation error should be chosen as small as possible, as ideally thinning should be done only at the end of the processing chain.

Merging. The meshes from different views are merged by pairs using local mesh operations like *vertex insertion*, *gap bridging* and *surface growth* (Fig. 17.5). Initially a master image is chosen. The other views are merged into it successively. Only those vertices are inserted whose absence would cause an approximation error bigger than a given threshold.

Final smoothing. Due to registration and calibration errors the meshes do not match perfectly. Thus, after merging the mesh is usually distorted and has to be be smoothed again.

Final mesh thinning. Mesh thinning is continued until the given approximation error is reached. For thinning purposes a classification of the surfaces according to curvature properties is also generated.

Geometrical mesh optimization. Thinning usually causes awkward distributions of the vertices, so that elongated triangles occur. Geometrical mesh optimization moves the vertices along the curved surface, in order to produce a better balanced triangulation.

Topological mesh optimization. At last the surface triangulation is reorganized using *edge swap* operations, in order to optimize certain criteria. Usually, the interpolation error is minimized (Fig. 17.6).

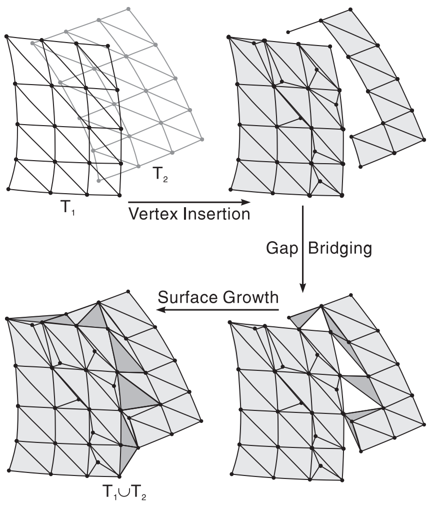


Figure 17.5: *Merging of the meshes using vertex insertion, gap bridging and surface growth operations.*

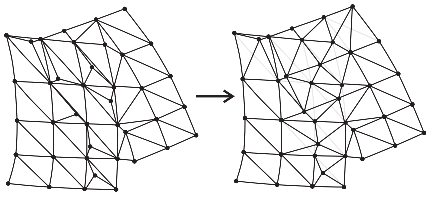


Figure 17.6: *Topological optimization of the mesh from Fig. 17.5 using edge swap operations. Triangles that are as equilateral as possible were the goal.*

The result of this process is a *mesh of curved triangles*. Our new modeling method is able to interpolate curved surfaces solely from the vertex coordinates and the assigned normal coordinates. This allows a compact description of the mesh, as modern data exchange formats like *Wavefront OBJ*, *Geomview OFF*, or *VRML*, support this data structure. The data may also be written in other formats like *AutoCad DXF* or *STL*. Currently no texture information is processed by the algorithm.

17.7 Surface modeling and smoothing

Many of the errors that are caused by the measuring process (*noise*, *aliasing*, *outliers*, etc.) can be filtered at the level of raw sensor data. A special class of errors (*calibration* and *registration errors*) first appears after merging the different views. We use a new modeling method that

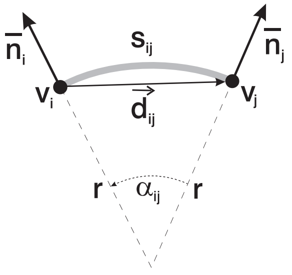


Figure 17.7: *Cross section* $\boldsymbol{s}$ *through a constantly curved surface.*

is based on the assumption that the underlying surface can be approximated by a *mesh of circular arcs* [13, 15] This algorithm allows the elimination of measuring errors without disturbing object edges. Filtering is done by first smoothing the normals and then using a *nonlinear geometry filter* to adapt the positions of the vertices.

17.7.1 Modeling of scattered data

In zero-order approximation we assume that the sampling density is high enough to neglect the variations of surface curvature between adjacent sample points. If this is true, the underlying surface can be approximated by a mesh of circular arcs. This simplified model provides a basis for all computations that our reverse engineering method requires, for example, *normal* and *curvature estimation*, *interpolation of curved surfaces*, or *smoothing of polyhedral surfaces*.

As an example we demonstrate how easy curvature estimation can be when this model is used. Figure 17.7 shows a cross section $\boldsymbol{s}$ through a constantly curved object surface between two adjacent vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$. The curvature c_{ij} of the curve $\boldsymbol{s}$ is ($c_{ij} > 0$ for concave and $c_{ij} < 0$ for convex surfaces)

$$c_{ij} = \pm\frac{1}{r} \approx \pm\frac{\alpha_{ij}}{d_{ij}} \approx \pm\arccos(\bar{\boldsymbol{n}}_i \cdot \bar{\boldsymbol{n}}_j) \tag{17.1}$$

which can be easily computed if the surface normals $\bar{\boldsymbol{n}}_i$ and $\bar{\boldsymbol{n}}_j$ are known. The *principal curvatures* $\kappa_1(i)$ and $\kappa_2(i)$ of $\boldsymbol{v}_i$ are the extreme values of c_{ij} with regard to all its neighbors $\boldsymbol{v}_j$:

$$\kappa_1(i) \approx \min_j(c_{ij}) \quad \text{and} \quad \kappa_2(i) \approx \max_j(c_{ij}) \tag{17.2}$$

The surface normals are computed separately, hence it is possible to eliminate noise by smoothing the normals without any interference of the data points. Therefore this method is much less sensitive to noise than the usual method for curvature estimation from sampled data, which is based on *differential geometry* [18].

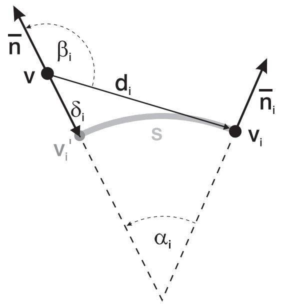


Figure 17.8: *Cross section* $\boldsymbol{s}$ *through a constantly curved surface. The position of vertex* $\boldsymbol{v}$ *is not measured correctly.*

17.7.2 Surface smoothing

This new approach can be used for smoothing of measuring errors with minimum interference of real object features like edges. If curvature variations of the sampled surface are actually negligible, while the measured data vary from the approximation of *circular arcs*, this must be caused by measuring errors. Therefore it is possible to smooth these errors by minimizing the variations.

For this purpose a measure δ is defined to quantify the variation of a vertex from the approximation model. Figure 17.8 shows a constellation similar to that of Fig. 17.7. Now the vertex $\boldsymbol{v}$ is measured at a wrong position. The correct position would be $\boldsymbol{v}_i'$ if $\boldsymbol{v}_i$ and the surface normals $\bar{\boldsymbol{n}}$ and $\bar{\boldsymbol{n}}_i$ match the simplified model perfectly (There exist different ideal positions $\boldsymbol{v}_i'$ for every neighbor $\boldsymbol{v}_i$). The deviation of $\boldsymbol{v}$ with regard to $\boldsymbol{v}_i$, given by

$$\delta_i \approx d_i \frac{\cos(\beta_i - \frac{\alpha_i}{2})}{\cos(\frac{\alpha_i}{2})} \tag{17.3}$$

can be eliminated by translating $\boldsymbol{v}$ into $\boldsymbol{v}_i'$. The sum over δ_i defines a cost function for minimizing the variations of $\boldsymbol{v}$ from the approximation model. Minimizing all cost functions of all vertices simultaneously leads to a mesh with minimum curvature variations for fixed vertex normals. Alternatively it is possible to define a *recursive filter* by repeatedly moving each vertex $\boldsymbol{v}$ along its assigned normal $\bar{\boldsymbol{n}}$ by

$$\Delta n = \frac{1}{2}\langle\delta_i\rangle = \frac{1}{2N}\sum_{i}^{N}\delta_i \tag{17.4}$$

where N is the number of neighbors of $\boldsymbol{v}$. After a few iterations all Δn converge towards zero and the overall deviation of all vertices from the circular arc approximation reaches a minimum. Restricting the translation of each vertex to only a single degree of freedom (direction of the

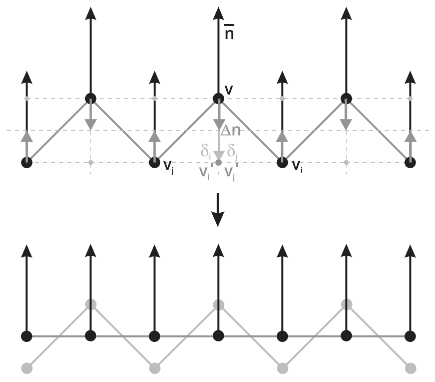


Figure 17.9: *Smoothing of an idealized registration error on a flat surface. After moving each vertex by* $0.5\langle\delta_i\rangle$ *along its normal vector, the data points are placed in the fitting plane of the original data set.*

surface normal) enables smoothing without seriously affecting small object details (compare with [14]). Conventional approximation methods, in contrast, cause isotropic daubing of delicate structures.

This procedure can be used for surface smoothing if the surface normals describe the sampled surfaces more accurately than the data points. In case of *calibration* and *registration errors* the previous assumption is realistic. This class of errors usually causes local displacements of the overlapping parts of the surfaces from different views, while any torsions are locally negligible. Oscillating distortion of the merged surface with nearly parallel normal vectors at the sample points are the result. Figure 17.9 shows an idealization of such an error if the sampled surface is flat. The upper and lower vertices derive from different images that were not registered perfectly. It is clear that this error can be qualitatively eliminated if the vertices are forced to match the defaults of the surface normals by translating each vertex by Eq. (17.4) along its normal. If the sampled surface is flat, the resulting vertex positions are placed in the fitting plane of the original data points. In the case of curved surfaces, the regression surfaces are curved too, with minimum curvature variations for fixed surface normals. This method generalizes standard methods for smoothing calibration and registration errors by locally fitting planes to the neighborhood of each vertex [11, 14]. Instead of fitting piecewise linear approximation surfaces, in our case surface patches with piecewise constant curvature are used.

From Fig. 17.9 the minimum sampling density for conserving small structures can be deduced. If the oscillations of the pictured data points are not caused by measuring errors but instead by the real structure of the sampled surface, this structure is lost when the normals are computed. This happens because the surface normals average over ±1 sampling interval. Therefore the sampling rate ν_n of the normals is only half the sampling rate ν_v of the vertices. In the depicted example ν_n is identical with the maximum space frequency of the sampled surface, which violates the *sampling theorem*. As a result, approximation of a mesh of circular arcs requires a sampling density that is at least four times higher than the smallest object details to be modeled. This means that the minimum sampling rate must be twice as high as the theoretical minimum given by the *Nyquist frequency*. Therefore further investigations are necessary to extend the new modeling method to higher orders of curvature variations, in order to get closer to the theoretical limit.

Figures 17.10 and 17.11 demonstrate that this method works well in case of registration and calibration errors. In case of noise or aliasing errors (*Moiré*) the surface normals are also distorted, but can simply be smoothed by weighted averaging. Thus filtering is done by first smoothing the normals (*normal filter*) and then using the described *geometry filter* to adapt the positions of the data points to these defaults.

17.7.3 Interpolation of curves and curved surfaces

Interpolation of curved surfaces (e. g., *curved triangles*) over n-polygons can simply be done by interpolation between circular arcs. For that purpose, a new surface normal for the new vertex is computed by linear interpolation between all surrounding normals. The angles between the new normal and the surrounding ones define the radii of the arcs as it is shown in Fig. 17.7 (for details, refer to Karbacher and Häusler [15]). Our method uses this simple interpolation scheme mainly for *geometrical mesh optimization*.

17.7.4 Experiments

In our experiments it turned out that the practicability of any surface reconstruction method depends strongly on the efficiency of its smoothing algorithms. Our method works best in case of registration and calibration errors. Figures 17.10 and 17.11 demonstrate that such errors in fact can be smoothed without seriously affecting any object details. The mesh of Fig. 17.10a was reconstructed from 7 badly matched range images. The mean registration error is 0.14 mm, the maximum is 1.5 mm (19 times the sampling distance of 0.08 mm!). The mean displacement of a single vertex by smoothing was 0.06 mm, the maximum was 0.8 mm (Fig. 17.10b). The displacement of the barycenter was 0.002 mm. This indicates that the smoothed surface is placed perfectly in the center of the difference volume between all range images.

In Fig. 17.11a the meshes from the front and backside of a ceramic bust do not fit because of calibration errors. The mean deviation is

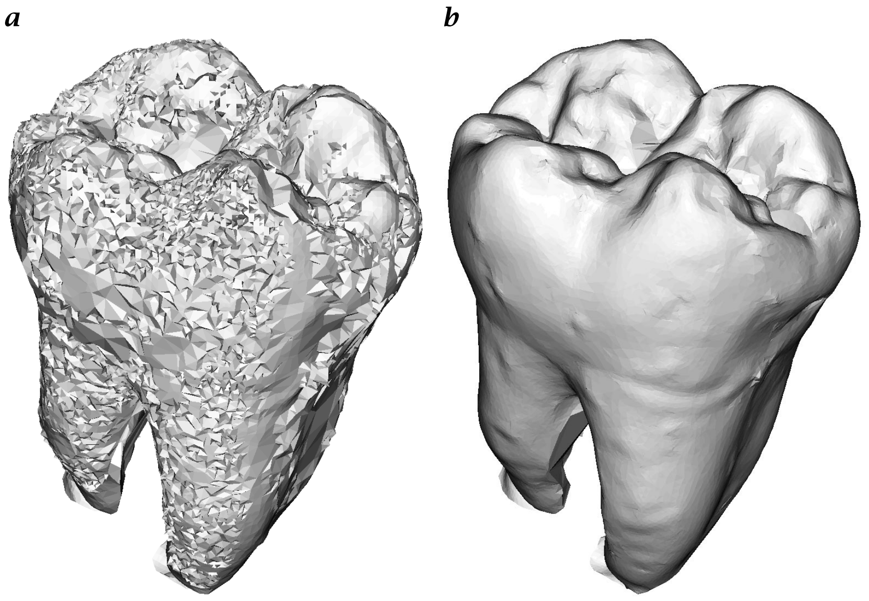


Figure 17.10: ***a*** *Distorted mesh of a human tooth, reconstructed from seven badly matched range images; and* ***b*** *result of smoothing.*

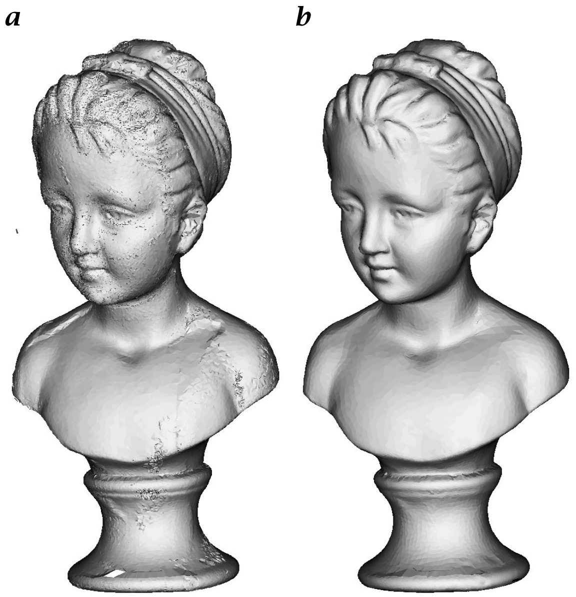


Figure 17.11: *Smoothing of a mesh containing calibration errors:* ***a*** *mesh after merging of 12 range images;* ***b*** *result of smoothing.*

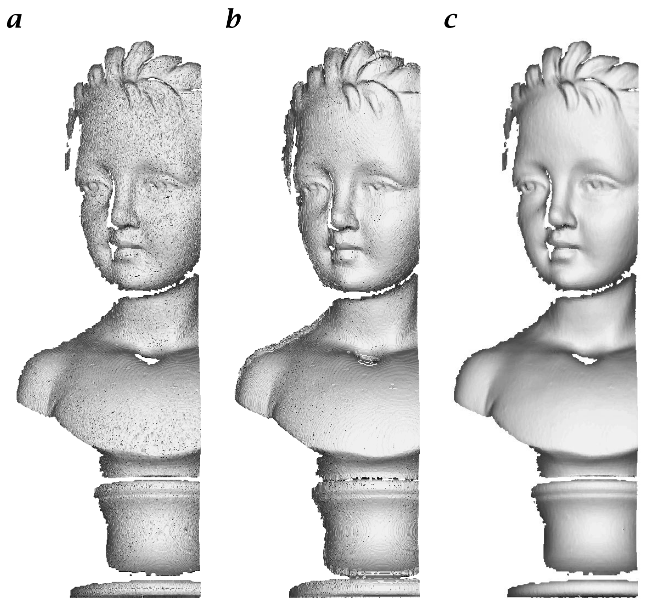


Figure 17.12: ***a*** *Noisy range image of the ceramic bust;* ***b*** *smoothed by a* 7×7 *median filter; and* ***c*** *by the new filter.*

0.5 mm, the maximum is 4.2 mm (the size of the bounding box is $11 \times 22 \times 8\ \mathrm{cm}^3$). The mean displacement by smoothing is 0.05 mm, the maximum is 1.3 mm (Fig. 17.11b). In this example the different meshes are not really merged, but solely connected at the borders, so that a displacement that was obviously smaller than the half of the distance between the meshes was sufficient.

Figure 17.12 demonstrates smoothing of measuring errors of a single range image (Figure 17.12c) in comparison to a conventional *median filter* (Figure 17.12b), which is the simplest and most popular type of edge preserving filters. Although the errors (variations of the smoothed surface from the original data) of the median filter are slightly larger in this example, the new filter shows much more noise reduction. Beyond that, the median filter produces new distortions at the borders of the surface. The new filter reduces the noise by a factor of 0.07, whereas the median filter actually increases the noise because of the produced artifacts. The only disadvantage of our filter is a nearly invisible softening of small details.

In all experiments the mean displacement of the data was smaller than the mean amplitude of the distortion, which shows that the errors introduced by the new filter are always less than the errors of the original data. In particular no significant global shrinkage, expansion or displacement takes place, a fact that is not self-evident when using

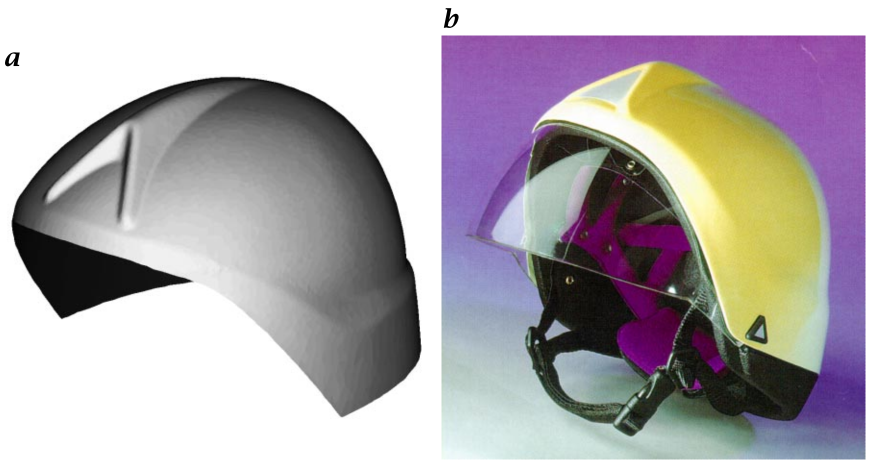


*Figure 17.13: **a** Reconstructed surface of a firefighter's helmet (/movies/17/helmet.mov) and **b** the helmet that was produced from the CAD data.*

real *3-D filters*. For realistic noise levels *edge preserving smoothing* is possible.

17.7.5 Results

In our experiments the errors that were reinduced by the modeling process were smaller than the errors in the original data (measuring, calibration and registration errors). The new smoothing method is specifically adapted to the requirements of geometric data, as it minimizes curvature variations. Undesirable surface undulations are avoided. Surfaces of high quality for *visualization*, *NC milling* and *"real" reverse engineering* are reconstructed. The method is well suited for *metrology* purposes, where high accuracy is desired.

17.8 Examples

We have tested our reverse engineering method by digitizing many different objects, technical as well as natural. The following examples were digitized by our sensors and reconstructed with our *SLIM*$^{3\text{-}D}$ software.

Figure 17.13 shows the results for a design model of a firefighter's helmet. The reconstructed CAD-data were used to produce the helmet (Fig. 17.13b). For that purpose, the triangular mesh was translated into a mesh of *Bézier triangles*,[2] so that small irregularities on the border

[2]This was done using software from the Computer Graphics Group.

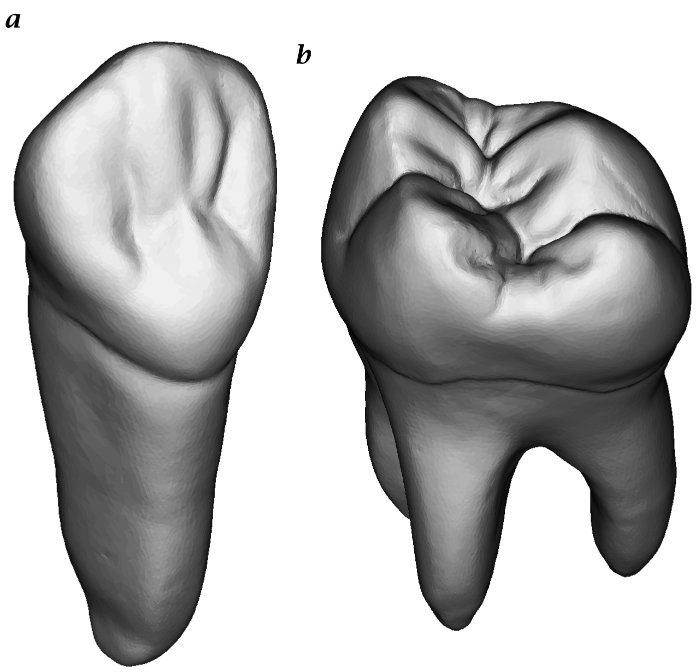


Figure 17.14: *Reconstructed surfaces of plaster models of* ***a*** *a human canine tooth (35,642 triangles, 870 kB); and* ***b*** *a molar (64,131 triangles, 1.5 MB); for 3-D models see* `/3dmodel/17/canine.wrl` *and* `/3dmodel/17/molar.wrl`*, respectively.*

could be cleaned manually. Eight range images containing 874,180 data points (11.6 MByte) were used for surface reconstruction. The standard deviation of the sensor noise is 0.03 mm (10% of the sampling distance), the mean registration error is 0.2 mm. On a machine with an Intel Pentium II 300 processor, the surface reconstruction took 7 min. The resulting surface consists of 33,298 triangles (800 kByte) and has a mean deviation of 0.07 mm from the original (unsmoothed) range data. For medical applications we measured a series of 26 human tooth models [19]. Figure 17.14 shows a reconstructed canine tooth and a molar. Ten views (between 13 and 14 MB) were processed. A rather complex object is shown in Fig. 17.15. It is the reconstruction of a console of an altar, digitized for the *Germanisches Nationalmuseum*. Twenty views were merged to cover the ancient, rough surface in detail. Problems during digitizing arose due to gold plating and different kinds of paint.

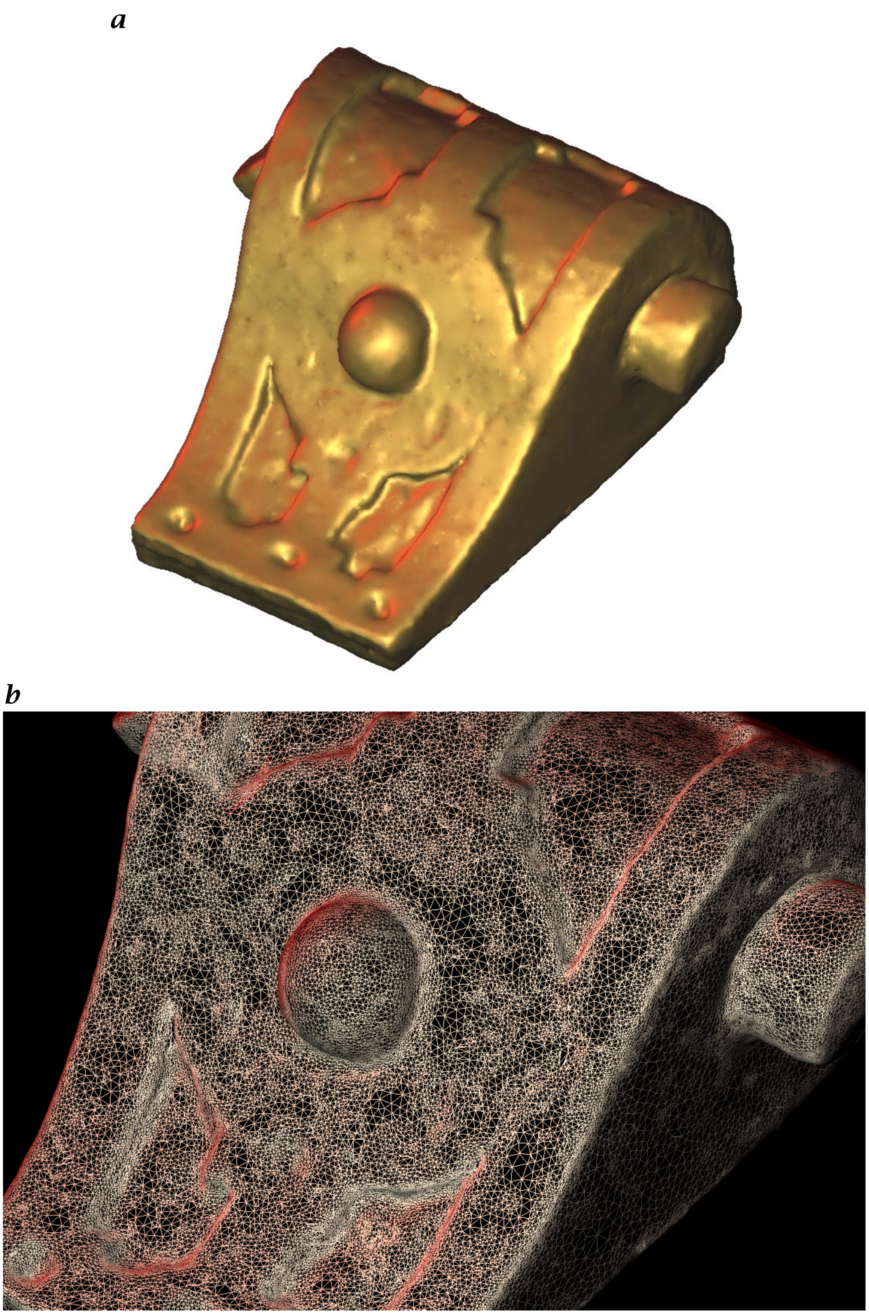


Figure 17.15: *Console of an altar:* ***a*** *rendered surface;* ***b*** *zoom into wire frame (193,018 triangles); for 3-D model see* `/3dmodel/17/console.wrl`.

17.9 Conclusions

We have presented a nearly automatic procedure for digitizing the complete surface of an object and for reconstructing a triangular mesh with high accuracy for metrology purposes. Calibration, registration and smoothing errors are usually less than sensor noise. The user defines a limit for the approximation error, so that the density of the final mesh is specified.

Acknowledgment

Parts of this work were supported by the "Deutsche Forschungsgemeinschaft" (DFG) #1319/8-2 and by the "Studienstiftung des deutschen Volkes."

17.10 References

[1] Häusler, G., Schönfeld, H., and Stockinger, F., (1996). Kalibrierung von optischen 3D-Sensoren. *Optik*, **102**(3):93–100.

[2] Johannesson, M., (1995). Calibration of a MAPP2200 sheet-of-light range camera. In *Proc. of the SCIA*. Uppsala.

[3] Ritter, D., (1996). *Merkmalsorientierte Objekterkennung und -lokalisation im 3D-Raum aus einem einzelnen 2D-Grauwertbild und Referenzmodellvermessung mit optischen 3D-Sensoren*. Dissertation, Friedrich Alexander University of Erlangen.

[4] Brunnström, K. and Stoddart, A. J., (1996). Genetic algorithms for free-form surface matching. In *14th Int. Conference on Pattern Recognition*. Vienna, Austria.

[5] Stoddart, A. J. and Brunnström, K., (1996). Free-form surface matching using mean field theory. In *British Machine Vision Conference*. Edinburgh, UK.

[6] Stoddart, A. J. and Hilton, A., (1996). Registration of multiple point sets. In *14th Int. Conference on Pattern Recognition*. Vienna, Austria.

[7] Besl, P. J. and McKay, N. D., (1992). A method of registration of 3-D shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **14**(2):239–256.

[8] Edelsbrunner, H. and Mücke, E. P., (1994). Three-dimensional alpha shapes. *ACM Transactions on Graphics*, **13**(1):43–72.

[9] Veltkamp, R. C., (1994). *Closed Object Boundaries from Scattered Points*, Vol. 885 of *Lecture Notes in Computer Science*. Berlin, Heidelberg, New York: Springer Verlag.

[10] Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., and Stuetzle, W., (1994). Piecewise smooth surface reconstruction. In *Proceedings, SIGGRAPH '94, Orlando, Florida, July 24–29, 1994*, A. Glassner, ed., pp. 295–302. Reading, MA: ACM Press.

[11] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W., (1992). Surface reconstruction from unorganized points. In *Proceedings, SIGGRAPH '92, Computer Graphics, Chicago, Illinois; 26–31 July 1992*, E. E. Catmull, ed., Vol. 26, pp. 71–78.

[12] Curless, B. and Levoy, M., (1996). A volumetric method for building complex models from range images. In *Proceedings, SIGGRAPH 96 Conference, New Orleans, Louisiana, 04–09 August 1996*, H. Rushmeier, ed., Annual Conference Series, pp. 303–312. Reading, MA: Addison Wesley.

[13] Karbacher, S., (1997). *Rekonstruktion und Modellierung von Flächen aus Tiefenbildern*. Dissertation, Friedrich Alexander University of Erlangen, Aachen: Shaker Verlag.

[14] Turk, G. and Levoy, M., (1994). Zippered polygon meshes from range images. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, A. Glassner, ed., pp. 311–318. Reading, MA: ACM Press.

[15] Karbacher, S. and Häusler, G., (1998). A new approach for modeling and smoothing of scattered 3D data. In *Three-Dimensional Image Capture and Applications, 01/24 – 01/30/98, San Jose, CA*, R. N. Ellson and J. H. Nurre, eds., Vol. 3313 of *SPIE Proceedings*, pp. 168–177. Bellingham, Washington: The International Society for Optical Engineering.

[16] Lounsbery, M., (1994). *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, University of Washington.

[17] Taubin, G., (1995). A signal processing approach to fair surface design. In *Proceedings, SIGGRAPH 95 Conference, Los Angeles, CA, 6–11 August 1995*, R. Cook, ed., Annual Conference Series, pp. 351–358. Reading, MA: Addison Wesley.

[18] Besl, P. J. and Jain, R. C., (1986). Invariant surface characteristics for 3-D object recognition in Range Images. *Computer Vision, Graphics, and Image Processing*, **33**:33–80.

[19] Landsee, R., v. d. Linden, F., Schönfeld, H., Häusler, G., Kielbassa, A. M., Radlanski, R. J., Drescher, D., and Miethke, R.-R., (1997). Die Entwicklung von Datenbanken zur Unterstützung der Aus-, Fort- und Weiterbildung sowie der Diagnostik und Therapieplanung in der Zahnmedizin—Teil 1. *Kieferorthopädie*, **11**:283–290.

18 Topographical Maps of Microstructures Generated by Depth-from-Focus Techniques

Torsten Scheuermann[1], Georg Wiora[2], and Matthias Graf[3]

[1]Fraunhofer USA Inc., Ann Arbor (MI), USA
[2]Daimler Chrysler AG Forschungszentrum, Ulm, Germany
[3]Fraunhofer ICT, Pfinztal, Karlsruhe, Germany

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

18.1 Introduction

The measurement of surface topography in the submicron range has become more important since synthetic microsystems have assumed greater market penetration. Synthetic microsystems or microparts are used, for example, in ink jet printer devices, as an accelerometer for airbag systems, and in any microsized unit.

The production of these microelements requires high geometric accuracy. The development and the production process must be quality controlled, which includes the shape measurement of microstructures. In most cases it is sufficient to measure the profile of the surface or the so-called *height map* or *depth map* of the surface topography.

This chapter describes a simple optical approach allowing resolutions in optical (z-)axis direction with nanometer accuracy of objects with shining or mirror-like reflecting surfaces. A prototype apparatus impresses by robust functionality and was designed to use a common optical microscope as a base unit that is completed with easily purchased components including a computer-controlled object stage, a PC with video frame grabber and a CCD-camera. The approach works with a noncoherent white light source and uses the decrease of contrast and sharpness by defocusing to determine the surface topography. A great benefit is high accuracy through a large z-range. A digital image processing algorithm reconstructs the 2 1/2-D topography or coating thickness, respectively, of a wide range of shape kinds with high reliability. A fuzzy logic diagnostic system detects measuring errors by post-processing and assists nonexperienced users in interpreting the measurement results.

18.2 Depth-from-focus approaches

The so-called depth-from-focus approaches use the effects of diffraction when an optical system is defocused. The image loses its sharpness and contrast. If a single point of the image may be traced through the defocusing range you get an intensity curve where the maximum or minimum marks the best focus. In case of contrast detection you have to compare at least two image points, which represent a dark and a bright reflecting surface point. The intensity difference of the points could be interpreted as the contrast of these image points. In this case a maximum exists when the best focus is reached. The kind of signal tracing leads to different methods.

One kind of depth-from-focus approach investigates single points and scans the object surface laterally. For each point the best focus is found by a search algorithm. The focusing unit changes the focus permanently, controls the contrast or intensity of the signal simultaneously, and approaches with each step more and more to the best fit [1, 2]. The same principle is used by autofocus cameras, which gives the method its name: Autofocus method.

A further kind separates the defocusing range into several fixed steps. Each step produces one 2-D image. An array of all images is evaluated by post-processing through a computer. The correlation between contrast maxima location and topography of the object leads to a 2 1/2-D map. This is also the principle we use for the measuring approach presented here.

The second method was originally used in confocal microscopy. However, the main users, for example, biologists, are usually not interested in measuring the objects with high accuracy but in obtaining a sharp visualization. They need the capability of confocal microscopes to enhance lateral resolution and to visualize transparent objects in three-dimensionality [3, 4]. Furthermore, the microscopists would like to reduce the depth of focus, which depends on the aperture of the microscope objective. Uneven objects could lead to images with sharp and sharpless areas.

To avoid this disadvantage of light microscopy Agard [5], Wu and Schwarzmann [6] generated the so-called sharp image by inverse filtering of the image array. However, this method demands a very high computing power. Häusler [7] used another method. He just added the prefiltered images to the image array. This method produced a sharp image which seemed better than the originals of the image array. In Häusler and Körner [8] he described a much better working method to compose the sharp image by depuzzling the sharpest regions of each image. Pieper and Korpel [9], Sugimoto and Ichioka [10] and Itoh et al. [11] developed further methods to generate a sharp image.

The next innovation was to use this method not only to generate a sharp image but also to evaluate a topographic map of the object surface. The measuring of microscopic objects was described by Steurer et al. [12], Nayar and Nakagawa [13] and Nayar [14].

Engelhardt and Häusler [15] and Girod and Scherock [16] remarked that the application of depth-from-focus approaches is only successful when the object is structured and, therefore, projected patterns on the object surface. You may fancy an autofocus camera directed to a white wall. The focusing will fail because the camera needs dark and bright image points to evaluate the contrast. Their applications referred to measuring of larger objects.

Scheuermann et al. [17] described the use of random pattern illumination to enhance the contrast. We developed a regression method to interpolate z-values, which may be between the moving steps of the object stage [18]. Scheuermann [19] gives a fundamental description of the optical theory, the comparison of random and periodic illumination patterns like stripes and checkers, the interpolating algorithm, a

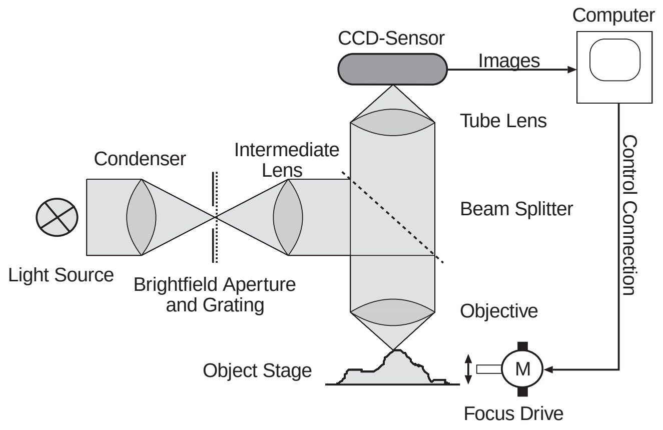


***Figure 18.1:** The main components of the surface measurement system: reflection microscope, structured grating, CCD, focus drive and computer.*

fuzzy-logic-based diagnostic system to assist the interpreting of measuring results. It was based on several master's theses from Hempel [20], Pfundt [21], and Wagner [22]. Graf [23] extended the method to determine the thickness of transparent layers.

The depth-from-focus approaches belong to the class of triangulation methods for optical measurements of depth maps. A systematic overview of 3-D optical measuring techniques is given in Volume 1, Chapter 18. Häusler describes the limits of triangulation-based methods and gives an overview of interferometric approaches in Volume 1, Chapter 19.

18.3 System description

18.3.1 Setup

The system we developed consists of widely available components. As shown in Fig. 18.1, there is a slightly modified reflection microscope with a computer controlled focus and a video camera. Data acquisition and evaluation is controlled by a standard personal computer with a video frame grabber.

The only modification to the microscope is a structured grating, located at the bright field aperture. This is used to project a pattern on the object surface in the focus. The illumination optic closely follows the principle of Beyer and Riesenberg [24]. In this arrangement the objective is both projection and imaging optic. As a result, the scale of the projected pattern is independent of the objective magnification.

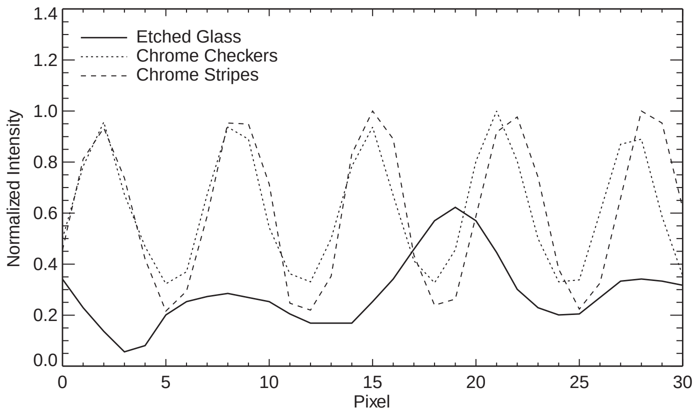


Figure 18.2: *Normalized intensity of different gratings. The stripe grating has a higher contrast as the checker grating at the same period length. The etched glass has been normalized over the whole image.*

Incoherent light is transmitted via the structured grating in an optical system. The surface of the object is at the same time the termination of the illumination chain and the beginning of the imaging optics. The light scattered or reflected by the object passes through another optical system to the CCD camera, where it is converted to an analog electrical signal. This signal is digitized by an ADC. Finally, the digital signal is processed by a computer using a digital contrast operator on the image matrix. The resulting contrast image is then used to reconstruct the surface.

One of the challenges of the technique is the manufacturing of the *gratings* used to generate the *projection pattern*. The usable diameter of the *bright field aperture* is only about 3 mm. This size is projected over the object on a camera with 760 pixels. This leads to a medium structure size of about 4 μm. Because the camera usually does an oversampling of the optical resolution, the practical structure size is about 8 μm, the minimum spatial wavelength is 16 μm. Figure 18.2 shows some intensity plots of some grating types. *Slide gratings*, *etched glass grating*, and *chrome grating* were investigated.

The slide gratings can be ruled out because of their poor quality. At first glance the etched glass seems to be more practical than the chrome gratings. A closer look at the measurement results with these two grating types shows that the chrome grating gives much better results, especially on well-defined reflecting surfaces, while on stochastic surfaces the difference is not so large.

One reason for this is that the surface of etched glass is not planar. A simple model for this kind of surface would be an arrangement of microlenses. You can see that this results in a local focus deviation in the projected pattern. This focus deviation results in an error in height measurement. The regular chrome gratings with a striped pattern provided the best measurement results, especially on reflecting surfaces like silicon microdevices or integrated circuits. To obtain the highest possible contrast, the period of the grating should be a multiple of the camera pixel period.

18.3.2 Measuring procedure

The measuring procedure includes the following steps:

1. Setting the focus range. This could be done automatically, but the user may more easily select the interesting *focus range* himself. This range is subdivided into steps depending on the desirable axial resolution and the objective.
2. The focus drive moves the stage to the first position.
3. An image is acquired. The contrast operator is applied to the image.
4. The data is processed in a sequential algorithm (Section 18.5.2).
5. The stage position is increased by the stepsize.
6. Repeat from the second step until the focus range has been scanned.
7. Post-processing and display of result.

The result is a *depth map* of the surface shape and an additional image of the object that is sharp over the whole focus range.

18.3.3 Digital contrast operators

We required a contrast operator that gives an optimum response on focus series. However, what is the "best" contrast operator for this task? There is no general answer to this question, but the contrast operator is not a critical point of this measurement method as long as it meets the following criteria.

- The contrast calculation must not depend on the type of the projection pattern.
- It must work on regular as well as stochastic patterns.
- The contrast range should be positive.
- The noise sensitivity should be low.

The operator that best met these conditions in our experiments is the ConvSobel operator C. This operator is a modification of the Sobel operator (see Volume 2, Section 10.3.4) that uses a recursive binomial smooth filter on the result of the Sobel operator. This improves noise insensitiveness.

$$\boldsymbol{K}_1 = |\boldsymbol{G} * \boldsymbol{S}_1| + |\boldsymbol{G} * \boldsymbol{S}_2| \tag{18.1}$$

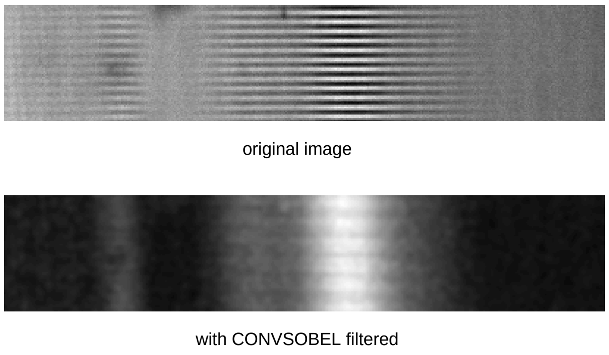


Figure 18.3: *The image of a stripe grid was partially focused on a mirror (above) and the ConvSobel operator C was applied on the resulting image (below). Bright areas show a high contrast.*

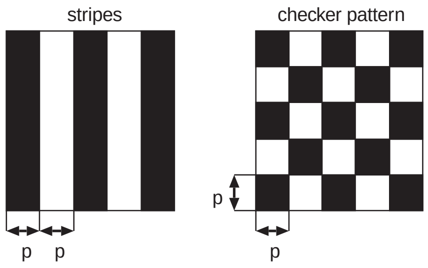


Figure 18.4: *The two examples of periodic illumination gratings: Stripe and checker pattern. The p defines the width or height of the pattern elements.*

$$\boldsymbol{K}_i = \boldsymbol{K}_{i-1} * {}^3\boldsymbol{R} \quad \text{with} \quad 2 \le i \le w \quad \text{and} \quad {}^3\boldsymbol{R} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{18.2}$$

w is the recursion depth (usually between 1 and 4). Figure 18.3 illustrates the application of this operator.

18.4 Optical theory

This section describes the optical transfer function (OTF) of an optical system that is defocused. The described OTF regards the diffraction of incoherent light. For depth-from-focus approaches it is interesting to determine how much image contrast is dependent upon the kind of active illumination. We chose two examples of periodic patterns to demonstrate the influence of dimensionality (stripe or checker) and periodicity on the image contrast.

In our theoretical model pattern content was either only totally transparent or totally nontransparent parts, respectively. Furthermore, we evaluate the influence of aberrations on the contrast value. This may help to decide which kind of illumination is more sensitive against aberrations. Finally, the results of this section help to find out which stepsize of the object stage would be optimum to reach a maximum z-resolution with the fewest steps possible.

To compare the different periodic patterns we define two characteristic values: The dimensionless contrast value $\tilde{C}$ and $\tilde{z}_{1/2}$. $\tilde{z}_{1/2}$ represents a dimensionless value of the defocusing distance in positive axial axis where the contrast curve reaches $\tilde{C} = 0.5\tilde{C}_{max}$. You may interpret $2\tilde{z}_{1/2}$ as the full-width-half-maximum (FWHm) of the contrast curve. $\tilde{C}_{max}$ is the dimensionless contrast value in the case of optimal focusing. The dimensionless contrast value itself is defined as

$$\tilde{C} = \frac{I_w - I_b}{I_w + I_b} \tag{18.3}$$

where I_w, I_b characterize the intensity values at the end of the optical system. The end of the microscope optical system is the target of a video device. The indices 'w' and 'b' mark the intensity of light that comes from the 'white' (transparent) or the 'black' (nontransparent) parts of the illumination grating (shown in Fig. 18.4). The surface of the sample should be a perfect plane mirror in this case. A dimensionless defocusing distance helps to compare different optical systems. We define a dimensionless axial axis with

$$\tilde{z} = \frac{8\pi}{\lambda} \sin^2\left(\frac{\alpha}{2}\right) z \tag{18.4}$$

where λ is the wavelength of a monochrome illumination, α is the aperture angle of the microscope objective, and z is a real distance value in object dimensions, which describes defocusing or movement of the object stage in optical axis direction. Origin of this axis is the point of best focusing. The formula was obtained by approximation of the optical phase defect by defocusing. The geometric relation was evaluated by Stokseth [25] and the approximation for low defocusing values was dealt with in [25, 26]. Also the pattern periodicity p can be defined as a dimensionless value. We define a dimensionless lateral axis with

$$\tilde{x} = \frac{2\pi}{\lambda} \sin(\alpha) x \tag{18.5}$$

where x is a real distance value in lateral object dimensions. In the case of the stripe pattern, x is the width p of the 'black' or 'white' stripes (see Fig. 18.4). Both are of equal width. In the case of the checker pattern both width and height are p. Using these definitions it is possible to describe $\tilde{C}$ and $\tilde{z}_{1/2}$ over $\tilde{p}$ for both pattern examples. This facilitates direct comparison of the different pattern examples.

18.4.1 Diffraction of incoherent light

Geissler describes the Fourier optics theory in Volume 1, Section 4.7. This background may be helpful in understanding both this and the next section. In Volume 2, Chapter 20 he details the depth-from-focus approach. The current section could be understood as a specific extension for the microscopic application focusing on high measuring accuracy.

The following diffraction model is based on the publications of Hopkins [27, 28] and Stokseth [25]. In this model a phase defect is responsible for a spherical aberration, the cause of images becoming sharpless if defocused. Hopkins [28] developed a very unwieldy equation representing the normalized, noncoherent optical transfer function (OTF) $\tilde{H}_{nc}$:

$$\begin{aligned}\tilde{H}_{nc}(\tilde{k},\tilde{z}) &= \frac{4}{\pi u}\left[\cos\left(\frac{u\tilde{k}}{2}\right)\left\{\beta J_1(u)\right.\right.\\ &+ \left.\sum_{n=1}^{\infty}(-1)^{n+1}\,\frac{\sin(2n\beta)}{2n}\left[J_{2n-1}(u)-J_{2n+1}(u)\right]\right\}\\ &- \left.\sin\left(\frac{u\tilde{k}}{2}\right)\sum_{n=0}^{\infty}(-1)^{n}\,\frac{\sin((2n+1)\beta)}{2n+1}\left[J_{2n}(u)-J_{2n+2}(u)\right]\right]\end{aligned} \tag{18.6}$$

J_i is the Bessel function of i-th order. We use the following additional variables to clarify the Hopkins equation:

$$u = \tilde{z}\tilde{k} \quad \text{and} \quad \beta = \arccos\left(\frac{\tilde{k}}{2}\right)$$

$\tilde{k}$ is the normalized spatial frequency. This modified equation is very important to attaining accurate results. It exists in several approximations, which lead mostly to nonprecise interpretations. In the case of optics for large objects they may be sufficient; however, in terms of microscopic optics with large apertures it is safer to use Eq. (18.6).

18.4.2 Active illumination with periodic patterns

Using Fourier optics we can easily calculate the light intensity difference of a bright (white) pattern element to its dark (black) neighborhood.

We define the transmittance of the illumination patterns in the case of stripes:

$$\tilde{\tau}_{st} = \operatorname{rect}\left(\frac{x}{p}\right) * \left[\sum_{n=-\infty}^{\infty}\delta(x-2np)\right] \tag{18.7}$$

and in the case of checkers:

$$\begin{aligned}\tilde{\tau}_{ch} &= \operatorname{rect}\left(\frac{x}{p},\frac{y}{p}\right) * \left[\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}\delta(x-2np)\delta(y-2np)\right. \\ &+ \left.\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}\delta(x-[1+2n]p)\delta(y-[1+2n]p)\right]\end{aligned} \tag{18.8}$$

The rect-function is defined as:

$$\operatorname{rect}(x,y) = \begin{cases} 1 & : \quad |x| \le 0.5 \wedge |y| \le 0.5 \\ 0 & : \quad \text{else} \end{cases} \tag{18.9}$$

The illumination and imaging optics of our microscope system are identical. For noncoherent light the intensity of a single point on the optical axis varies by defocusing. To obtain the contrast value we must define a contrast operator. In our case we used the normalized Laplacian operator. For theoretical evaluation it approximates the effects generated by our self-developed digital filter ConvSobel (see Section 18.3.3) sufficiently:

$$\hat{L}(\tilde{\boldsymbol{k}}) = -4 + 2\cos(\pi\tilde{k}_x) + 2\cos(\pi\tilde{k}_y) \tag{18.10}$$

However, it is necessary for the periodicity of the pattern elements to be identical with the optical periodicity of the camera target matrix. That means one pixel has to cover one illumination pattern element exactly. Otherwise this would lead to aliasing effects and contrast loss, both of which may influence accuracy.

After several steps we get a contrast formula for noncoherent light:

$$\tilde{C} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{p}^2 \operatorname{sinc}^2\left(\frac{\tilde{p}}{2}\tilde{k}_x\right)\operatorname{sinc}^2\left(\frac{\tilde{p}}{2}\tilde{k}_y\right)\hat{L}\tilde{H}_{nc}^2\tilde{T}d\tilde{k}_x d\tilde{k}_y \tag{18.11}$$

Using the Fourier transformation of $\tilde{\tau}$ we get for stripe pattern:

$$\tilde{C}_{st}(\tilde{z}) = \sum_{n=1}^{2\tilde{p}/\pi} \operatorname{sinc}^2\left(n\frac{\pi}{2}\right)(1-\cos(n\pi))\tilde{H}_{nc}^2\left(n\frac{\pi}{\tilde{p}},\tilde{z}\right) \tag{18.12}$$

or for checkers, respectively:

$$\begin{aligned}\tilde{C}_{ch}(\tilde{z}) &= \sum_{n=1}^{2\tilde{p}/\pi}\sum_{m=1}^{2\tilde{p}/\pi} \operatorname{sinc}^2\left(n\frac{\pi}{2}\right)\operatorname{sinc}^2\left(m\frac{\pi}{2}\right)[2-\cos(n\pi)-\cos(m\pi)] \\ & [1+\cos(\pi(n+m))]\tilde{H}_{nc}^2\left(\pi\sqrt{\frac{n^2}{\tilde{p}^2}+\frac{m^2}{\tilde{p}^2}}\tilde{z}\right) \\ &+ \sum_{n=1}^{2\tilde{p}/\pi}\operatorname{sinc}^2\left(n\frac{\pi}{2}\right)[1-\cos^2(\pi n)]\tilde{H}_{nc}^2\left(n\frac{\pi}{\tilde{p}},\tilde{z}\right)\end{aligned} \tag{18.13}$$

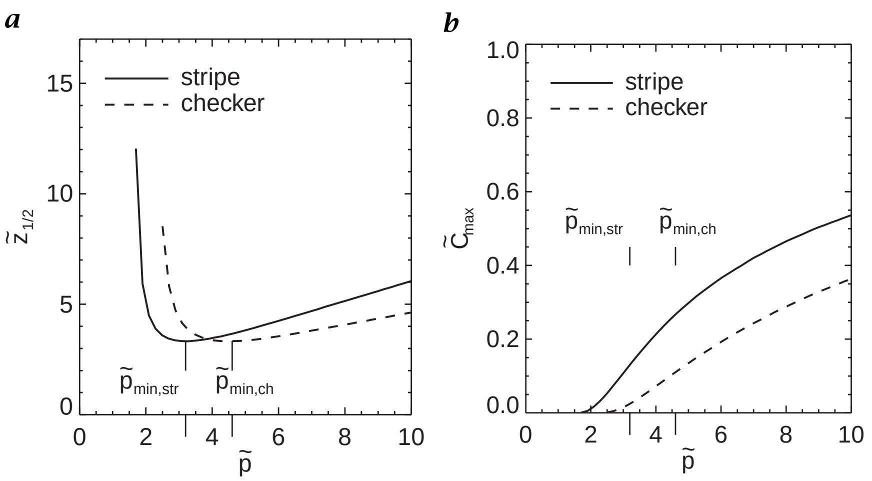


Figure 18.5: *The comparison of stripe and checker pattern:* ***a*** *$\tilde{z}_{1/2}$ over $\tilde{p}$;* ***b*** *the contrast range $\tilde{C}_{max}$ over $\tilde{p}$.*

The preceding equations have to be evaluated numerically. Figure 18.5 shows two diagrams explaining the difference between stripe and checker pattern.

The minima locations of the $\tilde{z}_{1/2}$ curves are different. The stripe pattern allows smaller $\tilde{p}$. That means that the dimension of the illumination pattern elements, the adapted dimension of the digital contrast filter, and the also adapted pixel dimension of the video device matrix can be decreased to enhance lateral resolution. The range between zero and $\tilde{p}_{min}$ would not provide more benefits because the optical resolution decreases enormously.

The second diagram shows the $\tilde{C}_{max}$ curves (contrast range) over $\tilde{p}$. We can also demonstrate that a stripe pattern has greater benefits than a checker pattern. The contrast range is much higher, which is very important for measuring accuracy. Better contrast increases the *signal-to-noise ratio* (SNR), which is responsible for the measurement resolution of our system (see Section 18.5.4 and Fig. 18.2).

We use the value of $\tilde{z}_{1/2}$ to evaluate the best axial stepsize of the object stage when we take an image sequence. The stepsize should be small enough to get at least three curve points around the contrast maximum within a range of $2z_{1/2}$. If we know p, the magnification, and the aperture of the microscope optic we are able to evaluate $z_{1/2}$. In Section 18.5.3 we demonstrate that the FWHM $2z_{1/2}$ divided by the number of regression points minus one represents the best stepsize.

18.4.3 Aberrations

From the Seidel's theory we get an amplitude transfer function H including terms describing the three kinds of aberration: astigmatism, coma, and spherical aberration.

The following equation was described by Wilson and Carlini [29]:

$$H = \exp \frac{1}{2} i\tilde{z}\tilde{k}^2 \exp\left(2\pi \mathrm{i}[A\tilde{k}^4 + B\cos(\phi\tilde{k}^3) + C\cos^2(\phi\tilde{k}^2)]\right) \operatorname{circ}(\tilde{k}) \tag{18.14}$$

The constants A, B, and C characterize the intensity of spherical aberration, coma, and astigmatism. We found out that the stripe pattern is more robust against aberrations that usually appear in microscope objectives.

Another typical aberration is field curvature. An even plane like a mirror may be measured and would appear to have spherical topography. This aberration has to be eliminated by subtracting a reference measurement. Reference could be the topography of the even mirror.

The preceding aberrations are based on monochromatic illumination. Chromatic aberrations could influence the contrast curve as well. The contrast curve distorts when we use polychromatic light. It is clear that this effect may result in less accuracy because the symmetrically working regression algorithm cannot find the real maximum of $\tilde{C}$.

18.5 Reconstruction of topography

The reconstruction method has a simple algorithmic base that has been developed to use as much information of the raw data as possible. Common limitations like memory and time restrictions had to be taken into account and influenced the method chosen.

18.5.1 Boundary conditions

The algorithm should reconstruct a surface topography from a focus series with an "unlimited" number of frames using only a few megabytes of memory and a few minutes on a personal computer (Pentium, 100 MHz). A sequential algorithm is an elegant solution for this problem because it does not require random access to the input data and makes on-line processing possible.

An important characteristic is that the stepsize Δz for the focus scan is fixed before the scan starts.

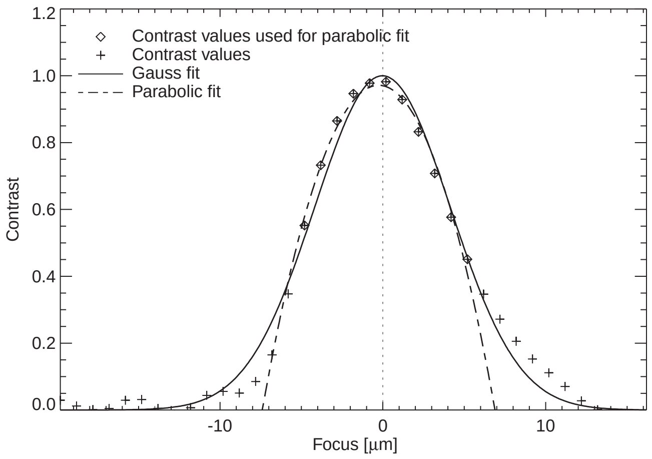


Figure 18.6: *Sample contrast curve. This curve was produced with a* 10× *Epiplan objective and a green filter to reduce axial chromatic aberration.*

18.5.2 Regression algorithm

The algorithm is based on an approach similar to that of [13]. The idea is to compute the local image contrast for each pixel, while scanning the focus range. A sample contrast curve for one pixel is shown in Fig. 18.6. With simple detection of the maximum location, a discrete topographic map is obtained.

This approach does not use the full information that is contained in the contrast curve. By fitting a function on the discrete data, localization of the maximum can be improved.

The most obvious fit function seems to be a Gaussian with an offset:

$$k(z) = a_0 + a_1 e^{-(z-z_M)^2/a_2} \tag{18.15}$$

For this kind of function a regression is not possible. A (slow) numerical optimization method had to be used and random access to all data would be necessary.

A closer look reveals that most of the information is contained in the planes next to the focus plane. Therefore, the contrast function can be approximated by a parabola:

$$k(z) = a_0 z^2 + a_1 z + a_2 \tag{18.16}$$

The maximum position of a parabola, which is given by

$$z_M = -\frac{a_1}{2a_0} \tag{18.17}$$

is easy to compute using the following regression formula:

$$z_M = \frac{1}{2}\frac{\overline{kz_2}\,(\bar{z}_3 - \bar{z}_1\bar{z}_2) + \overline{kz}\left(\bar{z}_2^2 - \bar{z}_4\right) + \bar{k}\,(\bar{z}_1\bar{z}_4 - \bar{z}_2\bar{z}_3)}{\overline{kz_2}\left(\bar{z}_2 - \bar{z}_1^2\right) - \overline{kz}\,(\bar{z}_3 - \bar{z}_1\bar{z}_2) + \bar{k}\left(\bar{z}_1\bar{z}_3 - \bar{z}_2^2\right)} \tag{18.18}$$

The following abbreviations have been used (the index of the maximum element is L_reg):

$$\begin{aligned}
\bar{z}_1 &= \frac{1}{2L_\text{reg}+1}\sum_{l'=0}^{2L_\text{reg}} z_{l'} & \bar{z}_2 &= \frac{1}{2L_\text{reg}+1}\sum_{l'=0}^{2L_\text{reg}} z_{l'}^2 \\
\bar{z}_3 &= \frac{1}{2L_\text{reg}+1}\sum_{l'=0}^{2L_\text{reg}} z_{l'}^3 & \bar{z}_4 &= \frac{1}{2L_\text{reg}+1}\sum_{l'=0}^{2L_\text{reg}} z_{l'}^4 \\
\bar{k} &= \frac{1}{2L_\text{reg}+1}\sum_{l'=0}^{2L_\text{reg}} K_{l'} & \overline{kz} &= \frac{1}{2L_\text{reg}+1}\sum_{l'=0}^{2L_\text{reg}} K_{l'}z_{l'} \\
\overline{kz_2} &= \frac{1}{2L_\text{reg}+1}\sum_{l'=0}^{2L_\text{reg}} K_{l'}z_{l'}^2
\end{aligned} \tag{18.19}$$

The convergence and numerical accuracy of Eq. (18.18) is much better if the z-offset of the maximum element is shifted to zero for the regression and only added to z_M after the regression.

The algorithm computes the regression moments $\bar{k}$, $\overline{kz}$, $\overline{kz_2}$ in the neighborhood of the maximum. For this, the z-stepsize and the approximate width of the contrast curve must be known in advance. These depend only on the objective used (see Section 18.4.2).

The results are acceptable if the global maximum of the contrast curve is used as z location for the regression, but some systematic errors at object edges can be reduced if the sharpest local maximum is used instead. As criterion μ for sharpness, the sum of the differences between the maximum contrast value and its predecessor and successor is used. This is similar to the second deviation of the contrast curve.

18.5.3 Optimum stepsize

As stated in Section 18.5.1 the stepsize for the focus scan has to be determined before it starts. Because the stepsize is evident for both resolution and total computation time it is worth considering how to choose an optimum stepsize.

The z range that can be used for the parabola regression should include only values from the top half of the contrast curve, because the shape differs too much from a parabola below. The optimum stepsize,

Table 18.1: *Typical stepsizes, $z_{1/2}$, for usual objectives calculated with Eq. (18.21) and noise-limited resolutions δz from Eq. (18.32). The measurement noise is assumed to be 7 % ($\varepsilon = 0.07$). The stripe width p'' on the grating is approximately 12 μm corresponding to $p' \approx 24\,\mu$m (three camera pixels). The parameters of Epiplan 100× are outside the valid range of the approximation formula.*

Objective	$z_{1/2}$ [μm]	δz [nm]
Epiplan 10×/0.3	5.25	692
Epiplan 20×/0.5	1.59	210
Epiplan 50×/0.75	0.457	60
Epiplan 100×/0.9	(0.255)	34

Δz_{opt}, can then easily be calculated from the number of planes to use for the regression L_{reg} and the FWHM $2z_{1/2}$:

$$\Delta z_{\text{opt}} = \frac{z_{1/2}}{L_{\text{reg}} - 1} \tag{18.20}$$

Scheuermann [19] found an approximation formula for estimating the FWHM of the contrast curve when illuminating with a stripe grating. With this estimation, the optimum stepsize for a stripe grating is

$$\Delta z_{\text{opt}} = \begin{cases} \dfrac{1}{2}\dfrac{0.225p\sin\alpha + 0.13\lambda}{(L_{\text{reg}} - 1)\sin^2\frac{\alpha}{2}} & : \quad p\sin\alpha > 3.87 \\[2ex] \dfrac{0.13\lambda}{(L_{\text{reg}} - 1)\sin^2\frac{\alpha}{2}} & : \quad 3.2 \le p\sin\alpha \le 3.87 \end{cases} \tag{18.21}$$

with the stripe width p in object coordinates. α is the objective aperture angle. Table 18.1 shows some typical stepsizes.

18.5.4 Noise-limited resolution

Finding the maximum of a curve with subpixel accuracy is a very common problem. A continuous function has to be sampled in discrete steps. The maximum position of the discrete data is determined by fitting a function. Two important questions show up: What stepsize is optimal and how accurate can the maximum be determined? We used a simple model to investigate these questions theoretically:

- The continuous function $f(z)$ is a Gaussian that is sampled with three equidistant points $f_i = f(z_i)$, $i \in [1..3]$. The stepsize is $\Delta z = z_i - z_{i-1}$.
- The fit function is a parabola.

- The locations of the sample points z_i have a small error compared to the error $\delta f_i = \varepsilon f_i$ of the sample values f_i.

Two extreme cases are obvious:

1. The middle sample hits exactly the maximum of the Gaussian (best case).
2. Two samples are equidistant to the left and right of the maximum, the third lays beside (worst case).

Error propagation. The maximum location of the parabola is given by a quadratic interpolation:

$$z_M = \frac{\Delta z}{2} \frac{f_3 - f_1}{2f_2 - f_1 - f_3} \tag{18.22}$$

With a linear error propagation for δf_i, the error δz_M for the maximum location is:

$$\delta z_M = \sqrt{2}\varepsilon\Delta z \frac{\sqrt{f_2^2 f_3^2 - f_2^2 f_1 f_3 - f_2 f_3^2 f_1 + f_1^2(f_2^2 - f_2 f_3 + f_3^2)}}{(f_1 - 2f_2 + f_3)^2} \tag{18.23}$$

The function value for f_i is given by a Gaussian with FWHM

$$b = 2z_{1/2}$$

a maximum location z_0 and an amplitude F_0:

$$f_i = f(z_i) = F_0 2^{-4(z_i - z_0)^2/b^2} \tag{18.24}$$

With respect to the FWHM of the Gaussian we introduce a dimensionless stepsize:

$$\Delta\tilde{z} = \frac{\Delta z}{b} \tag{18.25}$$

Best case. In the best case, when $z_2 \equiv z_0$, Eq. (18.23) can be simplified by setting $f_1 \equiv f_3$. After inserting Eqs. (18.24) and (18.25) we obtain:

$$\delta z_M^{(b)} = \frac{\varepsilon\Delta\tilde{z}b}{2\sqrt{2}(2^{4\Delta\tilde{z}^2} - 1)} \tag{18.26}$$

The dimensionless error for the maximum location is:

$$\delta\tilde{z}_M^{(b)} = \frac{\delta z_M^{(b)}}{b} = \frac{\varepsilon\Delta\tilde{z}}{2\sqrt{2}(2^{4\Delta\tilde{z}^2} - 1)} \tag{18.27}$$

Worst case. The worst case is given by $z_2 = z_0 + \Delta z/2$. It follows that $f_2 \equiv f_1$. Inserting Eqs. (18.24) and (18.25) in Eq. (18.23) gives:

$$\delta z_M^{(w)} = \frac{\sqrt{2}\varepsilon\Delta\tilde{z}b}{1 - 2^{-8\Delta\tilde{z}^2}} \tag{18.28}$$

The dimensionless error for the maximum location is:

$$\delta\tilde{z}_M^{(w)} = \frac{\delta z_M^{(w)}}{b} = \frac{\sqrt{2}\varepsilon\Delta\tilde{z}}{1 - 2^{-8\Delta\tilde{z}^2}} \tag{18.29}$$

Optimum stepsize. The optimum stepsize can now be computed by setting the first deviation of Eqs. (18.27) and (18.29) to zero. For Eq. (18.27) this has no result, because $\delta\tilde{z}_M^{(b)}$ goes towards zero for large $\Delta\tilde{z}$. For Eq. (18.29) this leads to the equation

$$1 - 2^{8\delta\tilde{z}^2} + 16\delta\tilde{z}^2 \ln 2 = 0$$

The dimensionless optimum stepsize is:

$$\Delta\tilde{z}_o = 0.4760 \tag{18.30}$$

Resolution limit. With this value and Eq. (18.29) it is possible to calculate the best reachable dimensionless localization error dependent on the measurement noise ε:

$$\delta\tilde{z} = 0.9411\varepsilon \tag{18.31}$$

From this the possible resolutions for the microscopic topography measurement can be computed:

$$\delta z = 1.8822\varepsilon z_{1/2} \tag{18.32}$$

Table 18.1 shows some typical values for a realistic measurement noise of 7 %.

18.6 Systematic errors

This section describes typical systematic errors of the topographic measurement system.

18.6.1 Topographic edges

Topographic edges, like steps, cause aperture clipping when imaging an object point of the lower level. Because of that, points near a step are very dark in the microscopic image and the lateral resolution is reduced due to the smaller aperture.

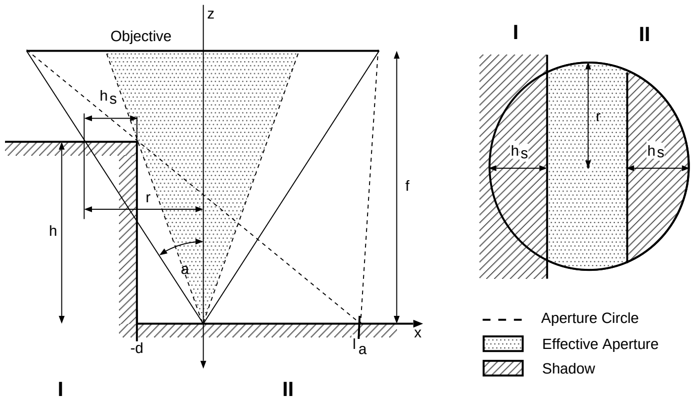


__Figure 18.7:__ Aperture clipping at an object step for a reflective object. For diffuse reflecting objects the aperture clipping is asymmetrical.

Assume that the objective is centered over a point P at $x = 0$ with the distance d to the edge, as shown in Fig. 18.7. Then the distance l_a up to that aperture clipping and, thus, resolution loss is given by:

$$l_a = \frac{\sin\alpha}{\sqrt{1 - \sin\alpha^2}} h \tag{18.33}$$

Inside this distance the intensity is reduced nearly linear to zero at $d = 0$. Obviously the aperture, $\sin\alpha$, is the most important factor. Smaller apertures are better for imaging deep holes. This is a common problem for all microscope types (even laser scanning microscopes).

Aperture clipping occurs on flat objects as well. A sloped plane mirror causes an asymmetric clipping with a loss of intensity and lateral resolution. If the slope angle is larger than the aperture angle α, no light reaches the objective.

18.6.2 Reflection edges

Not only topographic steps but also reflection edges cause problems. Figure 18.8 shows a simple model that explains the reasons: In focus position 2 a point in the dark object area is focused. A usual contrast maximum is found there. While moving the focus to position 3, a point is reached where the focus cone touches the bright object area. This induces a rise in the local contrast. A second contrast maximum results. The same happens when moving to position 1.

Watching the microscopic image while moving the focus over a reflection edge, the sharp bright edge moves into the dark area while blurring. This moving edge is recognized as a contrast maximum in different distances from the geometric edge depending on the focus

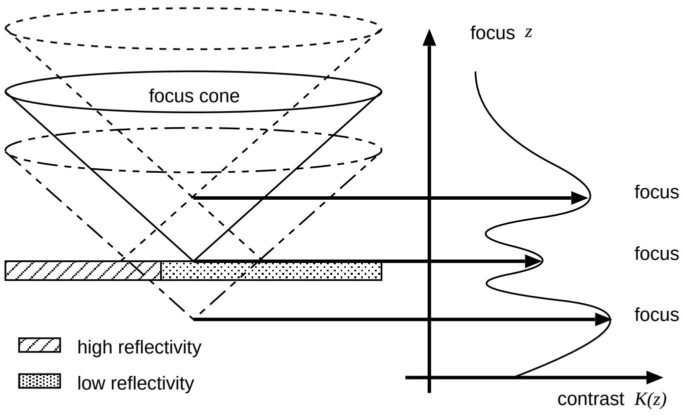


***Figure 18.8:** Geometric optic model for the origin of fake maxima in the contrast curve at reflection edges. The focus cones for different focus positions are shown. Focused above, on and below the surface.*

distance. Figure 18.9 shows an example. The profile across a thin aluminum layer on a silicon structure was measured. The contrast image shows clearly the high fake maximum running away from the edge while defocusing. The resulting profile shows a deep valley next to the edge, which makes the identification of the edge inaccurate.

18.6.3 Regression error

Another systematic error results from the different shapes of the regression parabola and the contrast curve. Between the best and worst case of sampling the contrast curve described in Section 18.5.4 the regression has a systematic deviation from the object shape. Figure 18.10 shows the profile of a slightly sloped planar mirror. The error is large when the object surface has the maximum distance to the focus planes and small if the surface is exactly in the focus plane.

On co-operative surfaces this error is the limiting factor for the system resolution! It is smaller when more planes are used for the quadratic regression. This requires a smaller z-stepsize and thus longer computation times.

18.6.4 Fuzzy logic error suppression

The automatic detection of measurement errors is a desirable feature for all kinds of measuring systems. It reduces the amount of knowledge that the user needs about possible errors and their causes. Basically, it is a pattern recognition problem. Neural networks are often successfully used for this task but too slow and too difficult to train.

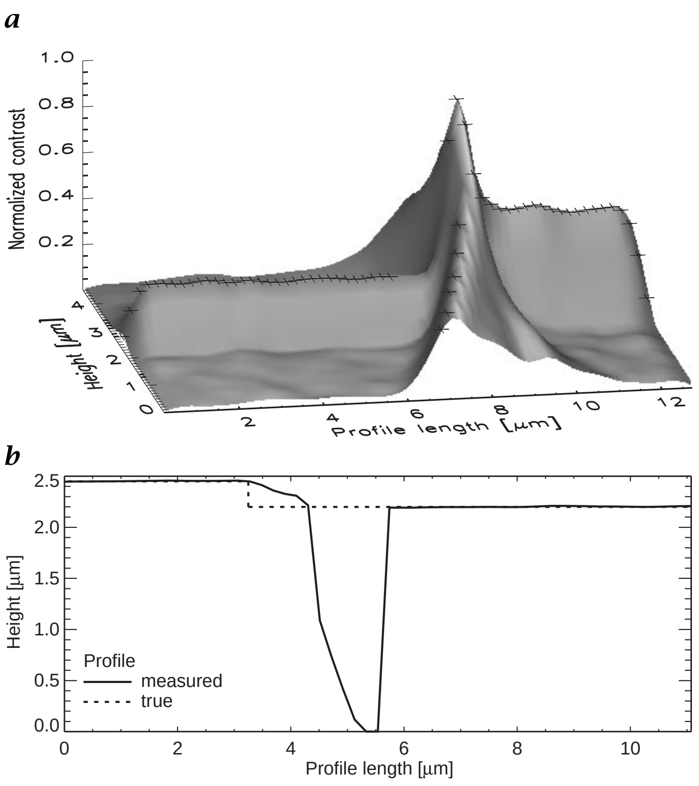


Figure 18.9: *In **a**, the contrast along a line crossing a reflectivity step on the object is shown for multiple focus planes. The maximum position for each pixel is marked with a cross. **b** shows the corresponding profile.*

When there is *expert knowledge* about the problem and it can be described with a few parameters and computation time is important, a fuzzy logic system recommends itself.

We used the following features as inputs for the fuzzy classificator:

Signal to average ratio: The signal to average ratio is a rough estimation of a contrast curve's quality. A high signal to average ratio suggests a low noise level and sharp maximum peak.

Median difference: The difference of the median filtered discrete height map $\boldsymbol{I}$ and the original map. As the median filter removes popcorn noise this feature responds well to outliers.

Maximum sharpness: Effects like those described in Sections 18.6.1 and 18.6.2 produce misleading maximums. These are usually not as sharp as a real focus maximum. The sharpness parameter is simply the sum of differences between the maximum contrast value

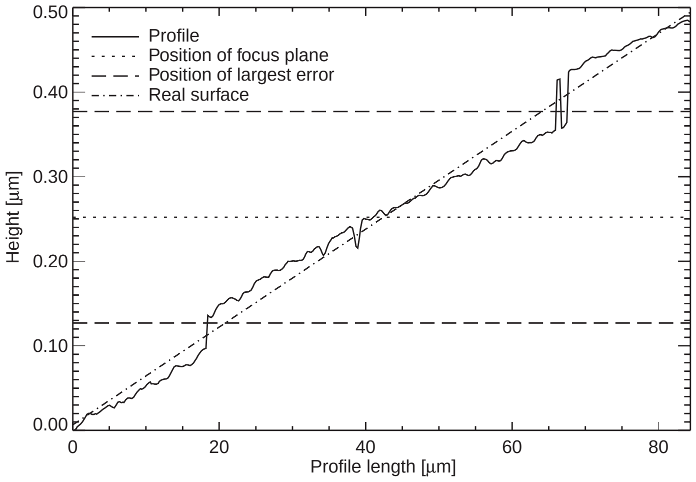


***Figure 18.10:** Measurement data of a planar slightly sloping surface. The points of the largest regression error are in a distance of* $\Delta z/2 = 0.125\,\mu m$ *to the measure point. The maximum error to the plane is about 25 nm. The slope angle is about 0.33°.*

and its predecessor and successor. This is an approximation for the second deviation of the contrast curve.

These variables are fuzzified and the fuzzy inference is calculated. The output of the fuzzy classificator is a robust quality measure in the range [0..1]. It is compared to a threshold for the decision about a points fate.

For further information on how fuzzy control works, see Chapter 22.

18.7 Measurement of layer thickness

Based on the method of map generation by depth from focus technique described here, we developed an application to measure layer thickness on transparent coated, reflective surfaces. The realization succeeded without any additional equipment, and most parts of the software could be utilized furthermore.

Transparent coated, reflective surfaces cause an additional maximum in the axial contrast response of the system, which correlates with the optical border between air and coating. Figure 18.11 shows a typical contrast curve for a transparent coated surface together with the theoretical values calculated by Eq. (18.44). For a more detailed examination, we modified the sequential algorithm to detect the two highest local maxima and calculate the appropriate regression moments.

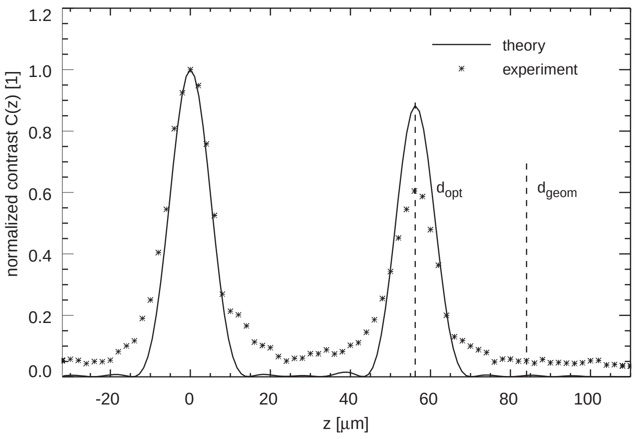


Figure 18.11: *Axial contrast values for a transparent coated silicon wafer, optical and geometrical layer thickness, d_{opt} and d_{geom}.*

The difference between the resulting height values z_1 and z_2 is called optical thickness

$$d_{opt} = |z_1 - z_2| \tag{18.34}$$

The geometrical thickness d_{geom} is a function of d_{opt}, of the refractive index n_c of the coating as well as of further system parameters.

18.7.1 Extensions of theoretical description

The different focus of the two surfaces and a further phase defect resulting out of the transparent coating are considered by the introduction of a separate OTF for each surface. The geometry of light refraction and phase defect is depicted in Fig. 18.12. Using Snells' law, the phase defect of the refracted wave is

$$w_n = d_{\text{geom}} \left(\sqrt{n_c^2 - \sin^2 \theta_a} - \cos \theta_a \right) \tag{18.35}$$

With the equation

$$\sin \theta_a = \tilde{k} \sin \alpha \tag{18.36}$$

we receive the phase defect as spatial frequency

$$w_n = d_{\text{geom}} \left(\sqrt{n_c^2 - \tilde{k}^2 \sin^2 \alpha} - \sqrt{1 - \tilde{k}^2 \sin^2 \alpha} \right) \tag{18.37}$$

The OTF of the coated surface is determined as

$$H_2 = \exp\left(2\pi \mathrm{i} \left(\lambda^{-1} \left(w_2 + w_n\right)\right)\right) \operatorname{circ} \left(\tilde{k}\right) \tag{18.38}$$

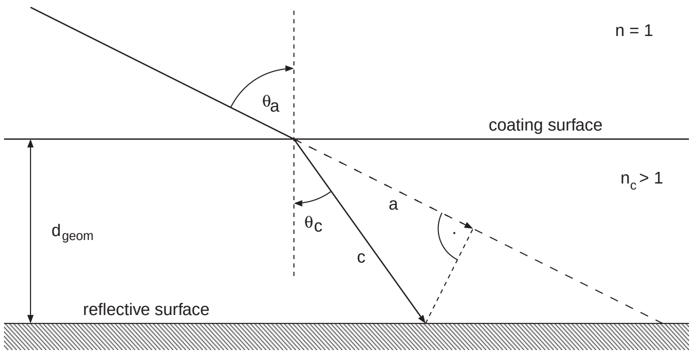


Figure 18.12: *Light refraction and phase defect at transparent layers.*

with

$$w_2 = 2\sin^2\left(\frac{\alpha}{2}\right)(z - d_{\text{geom}})\,\tilde{k}^2 \tag{18.39}$$

The relation of intensity amplitudes in case of transmission and reflection at an optical border are given by Fresnel's law for vertical and parallel polarized waves:

$$R_\perp = \left(\frac{n_c\cos\theta_a - \cos\theta_c}{n_c\cos\theta_a + \cos\theta_c}\right)^2 \qquad R_\parallel = \left(\frac{\cos\theta_a - n_c\cos\theta_c}{\cos\theta_a + n_c\cos\theta_c}\right)^2 \tag{18.40}$$

For the reflection of unpolarized light at the coating surface

$$\bar{R}_1 = \frac{1}{2}\left(R_\perp + R_\parallel\right) \tag{18.41}$$

and with ideal reflection at the coated surface we find

$$\bar{R}_2 = \frac{1}{2}\left((1 - R_\perp)^2 + (1 - R_\parallel)^2\right) \tag{18.42}$$

The normalized intensity in space frequencies is given by

$$\tilde{I} = \left(\bar{R}_1 H_1^2 + \bar{R}_2 H_2^2\right)\tilde{T} \tag{18.43}$$

The use of the Laplacian operator $\hat{L}$ and Fourier transformation leads to the normalized contrast depending on the focus position

$$\tilde{C}(z) = \int_\infty^\infty \int_\infty^\infty \tilde{p}^2 \operatorname{sinc}^2\left(\frac{\tilde{p}}{2}\tilde{k}_x\right)\operatorname{sinc}^2\left(\frac{\tilde{p}}{2}\tilde{k}_y\right)\hat{L}\tilde{I}d\tilde{k}_x d\tilde{k}_y \tag{18.44}$$

For known geometrical thickness and refractive index it is possible to determine d_{opt} as the difference between the local maxima positions

Table 18.2: *Approximation parameter x in Eq. (18.46) for some usual objectives*

Objective	Epiplan 10x/0.3	Epiplan 20x/0.5	Epiplan 50x/0.75	Epiplan 100x/0.9
κ [1]	0.045	0.103	0.050	0.101

of Eq. (18.44). We calculated a numerical variation of the parameters describing the layer and the optical system to find a more simple approximation with it. The result is a linear dependence of optical and geometrical thickness

$$d_{\text{geom}} = d_{\text{opt}}\gamma \tag{18.45}$$

For the correction factor γ there is a good approximation

$$\gamma = n_c + (n_c - 1)\kappa \tag{18.46}$$

The parameter κ must be calculated for each used objective but it is independent of the layer thickness and the refractive index n_c. Table 18.2 shows the values for some usual objectives.

The imaging with high aperture is especially sensitive to the spherical aberration caused by the transparent coating. Reference [29] deduced the relationship between the spherical aberration coefficient A in Eq. (18.14) and a parallel-sided glass plate.

$$A = \left(\frac{d_{\text{geom}}}{\lambda}\right)\left(\frac{n_c^3 - 1}{8n_c^3}\right)\sin^4\alpha \tag{18.47}$$

In our case the condition $A \leq 2$ represents the range of validity of Eq. (18.46). For the geometrical thickness that means

$$d_{\text{geom}} \geq \left(\frac{2\lambda}{\sin^4\alpha}\right)\left(\frac{8n_c^3}{n_c^3 - 1}\right) \tag{18.48}$$

Below this value, the correction factor γ must be calculated numerically with Eq. (18.44).

18.7.2 Experimental proof and resolution

For the experimental examination of the connection between optical and geometrical thickness, we have chosen transparent steps and flat ramps to determine both values simultaneously.

Obviously it is necessary to consider a computable distinctive mark between coated and noncoated surfaces. A simple way is to set a threshold of contrast relative to the main maximum. Its value depends on the

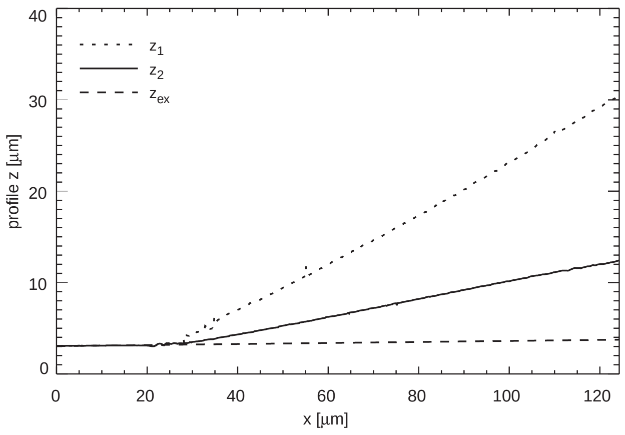


Figure 18.13: *Profile of a transparent ramp.*

reflection characteristic of the substratum and the coating surface itself. Although absorption and scattering inside of the coating material should be absent, sometimes these effects also influence the measurement and consequently the threshold.

After the determination of d_{opt} with Eq. (18.34), d_{geom} is calculated as the difference between z_1 and the extrapolated substratum surface in the coated area.

With this method, it is possible to examine transparent ramps, although they do not fit exactly to the theoretical given point of parallelism. Just the variation of thickness leads to a more detailed information on the correction factor γ.

Figure 18.13 shows the profile of a transparent, flat ramp. z_1 and z_2 describe the height values of the main and secondary contrast maximum, respectively, and z_{ex} the subtratum surface below the coating extrapolated with the data of the noncoated area. Also here it is necessary to set a threshold for the secondary contrast maximum.

The correction factor γ is nearly constant for each of the objectives, what is necessary for a significant measurement.

The border of a coating ramp is combined with an extreme small layer thickness. Here, we recognize the limit of resolution of the measurement system. In [23], there is a simple approach to deduce the theoretical resolution characteristic by superimposing two idealized Gaussian contrast curves. The minimum decomposable thickness d_{min} is proportional to the value of $\tilde{z}_{1/2}$ and the correction factor γ:

$$d_{\mathrm{min}} = \gamma \tilde{z}_{1/2} \left(\frac{\lambda}{8\pi}\right) \sin^{-2}\left(\frac{\alpha}{2}\right) \sqrt{\frac{1}{2\ln 2}} \qquad (18.49)$$

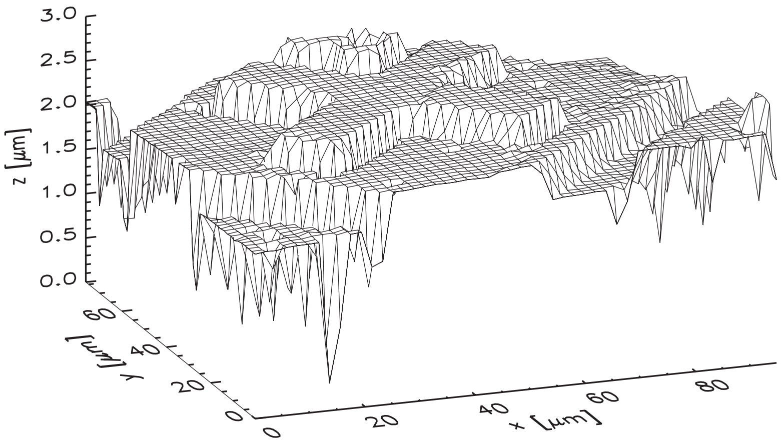


Figure 18.14: *A cross-like structure at the surface of an integrated circuit chip. The used objective had a magnification of 100 and an aperture of 0.9.*

For high aperture objectives and with it small $\tilde{z}_{1/2}$, the effect of particular interference gets important and disturbs the measurement. But in practice, here the double minimum–stepsize of the microscope table appears to be the bottom limit.

18.8 Applications

The following pictures demonstrate some applications of our approach. Figure 18.14 shows a topographic map of an integrated circuit. The object at the surface is a cross-structure. Using the coating measuring algorithm you would be able to measure thickness and shape of insulation and surface with the same measuring unit.

With LIGA technology (lithography by using synchrotron radiation, galvanic forming, and molding of plastics) it is possible to produce structures with large aspect ratio. The object shown in Fig. 18.15 is made of the transparent plastic PMMA (polymethylmethacrylate). The surface is shiny like glass. The measuring fails at the bottom of the quadratic blind holes. Because of the large aspect ratio of about 5 the intensity of the reflected light is not sufficient. Especially the edges at the bottom lie in the shadow of the illumination cone (see Fig. 18.7).

Figure 18.16 shows that our approach was quality controlled by a glass groove standard of the Physikalisch-Technische Bundesanstalt (PTB), Germany. The standard was checked with a reference measurement method. In this case PTB used a probe approach to specify the depth of the groove with 8.38 μm.

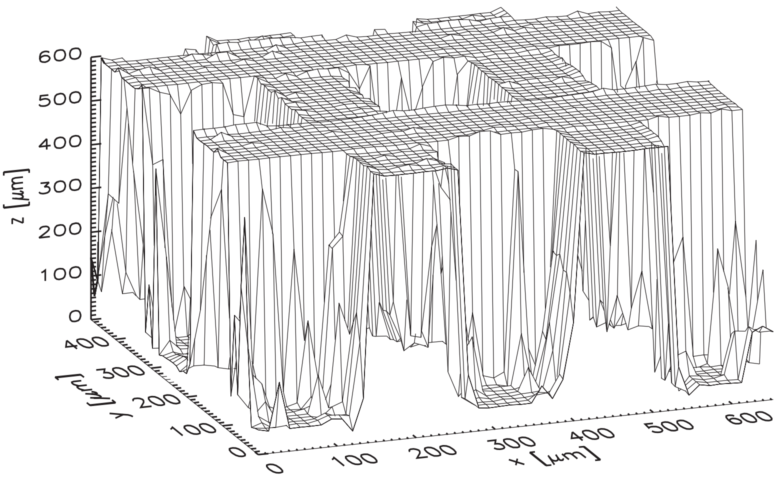


Figure 18.15: *A structure of polymethylmethacrylate. It was made by using the LIGA approach. The surface is shiny and the material is transparent like glass. You may recognize the aspect ratio of about 5. This leads to some artifacts at the bottom of the holes. The used objective had a magnification of 10 and an aperture of 0.3.*

18.9 Conclusions

Our goal was to design an optically shaped measuring unit with easily purchased components that can be successfully applied on synthetic or natural microscopic objects. The serious difficulty to measuring objects with large aspect ratio could be overcome in most cases. Of course restrictions exist in the case of undercut object parts or at steep surfaces. One reason is that this approach is only able to measure a topographic 2 1/2-D map from one viewing point. The other reason is that the reflected light may not be received if the surface gradient is too large. In this case a fuzzy logic diagnoses the problem areas that would lead to artifacts.

We could develop a theory to describe the optical effects. The contrast curve was evaluated theoretically. The results are identical with the theoretical expectations. To describe the optimal stepsize we use the FWHM of the evaluated contrast curve. We could also explain shadow effects at steps. Higher apertures leads to artifacts, especially in the region near or at surface steps or holes.

To get good contrast values we developed a special contrast operator based on a Sobel operator. However, the contrast operator is not the decisive factor in how accurately the approach works. Obviously it is the symmetry of the contrast curve and the right regression depth that lead to good results. Too many fitting points would lead to mea-

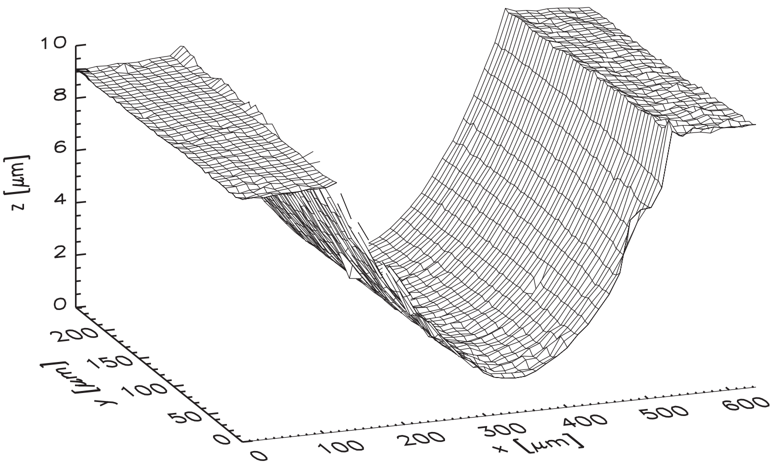


***Figure 18.16:** A glass groove standard of the Physikalisch-Technische Bundesanstalt (PTB), Germany. The original groove has a depth of 8.38 µm. The used objective had a magnification of 10 and an aperture of 0.3.*

suring errors because the shape of the used parabolic regression is only a rough approximation.

The comparison between stripe and checker illumination pattern showed that stripe patterns produce higher contrast. A large contrast range is an important condition for accurate measuring. Because of the higher SNR the regression algorithm computes better results. In the case of aberrations, the stripe pattern was much more robust than the checker pattern. The dimensionality of the pattern has to be small (stripes are one-dimensional) and the periodicity should be adapted on the periodicity of the video device target.

To measure the resolution we used a slanting plane mirror. We got an averaged error compared to a plane fit of 15 nm. However, it is clear that this result was measured under optimal conditions. It also depends on the optical properties of the microscope and on the surface properties of the object. A resolution below 100 nm should be real if an objective with high aperture is used.

If we use a thickness measuring unit of transparent coatings the optical situation is much more difficult. We developed an optical theory that allowed us to evaluate a resolution theoretically. Experiments and theory showed that coatings thinner than approximately 2 µm cannot be resolved. Also of complicated geometric design, artifacts can be produced by mirroring within the coating surfaces. Fortunately in most cases the insulation coatings we measure are usually thick enough and even.

18.10 References

[1] Brown, A. and Breitmeier, U., (1989). Industrial applications of an optical profilometer. *Proceedings of the SPIE*, **954**:200–207.

[2] Sheppard, C. J. R. and Matthews, H. J., (1988). The extended-focus, auto-focus and surface-profiling techniques of confocal microscopy. *Jour. Modern Optics*, **35**(1):145–154.

[3] Minsky, M., (1961). Microscopy apparatus: U.S. Patent No. 3013467. United States Patent Office.

[4] Sheppard, C. J. R. and Wilson, T., (1978). Depth of field in the scanning microscope. *Opt. Letters*, **3**(3).

[5] Agard, D. A., (1984). Optical sectioning microscopy: Cellular architecture in three dimensions. *Annual Review of Biophysics and Bioengineering*, **13**: 191–219.

[6] Wu, X. and Schwarzmann, P., (1993). 3D-Rekonstruktion mikroskopischer Bilder unter Berücksichtigung der Lichtabsorption. In *Proc. 15. DAGM Symp. 1993, Mustererkennung.*

[7] Häusler, G., (1972). A method to increase the depth of focus by two step image processing. *Optics Communications*, **6**(1):38–42.

[8] Häusler, G. and Körner, E., (1982). Expansion of depth of focus by image de-puzzling. In *IEEE/DAGM Proc., 6. Int. Conf. of Recogn., München.*

[9] Pieper, R. and Korpel, A., (1983). Image processing for extended depth of field. *Applied Optics*, **22**(10):1449–1453.

[10] Sugimoto, S. A. and Ichioka, Y., (1985). Digital composition of images with increased depth of focus considering depth information. *Applied Optics*, **24**(14):2076–2080.

[11] Itoh, K., Hayashi, A., and Ichioka, Y., (1989). Digitized optical microscopy with extended depth of field. *Applied Optics*, **28**(15):3487–3493.

[12] Steurer, J., Giebel, H., and Altner, W., (1986). Ein lichtmikroskopisches Verfahren zur zweieinhalbdimensionalen Auswertung von Oberflächen. In *Informatikberichte 125, 8. DAGM-Symp. Mustererkennung*, pp. 66–70. Berlin: Springer.

[13] Nayar, S. K. and Nakagawa, Y., (1990). Shape from focus: An effective approach for rough surfaces. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 218–225. Berlin, Heidelberg: Springer.

[14] Nayar, S. K., (1992). Shape from focus system for rough surfaces. In *DARPA, SC: Proceedings: Image understanding workshop.*

[15] Engelhardt, K. and Häusler, G., (1988). Acquisition of 3-D data by focus sensing. *Applied Optics*, **27**(22):4684–4689.

[16] Girod, B. and Scherock, S., (1989). Depth from focus of structured light. *Proceedings of SPIE*, **1194**:209–215.

[17] Scheuermann, T., Pfundt, G., Eyerer, P., and Jähne, B., (1995). Oberflächenkonturvermessung mikroskopischer Objekte durch Projektion statistischer Rauschmuster. In *Mustererkennung 1995—Verstehen akustischer und visueller Information*, G. Sagerer, S. Posch, and F. Kum-

mert, eds., Vol. 17 of *Informatik aktuell*, pp. 319–326, DAGM. Berlin, Heidelberg, New York: Springer.

[18] Pfundt, G. and Scheuermann, T., (1996). Eine einfache Methode zur mikroskopischen Gestaltvermessung und tiefenscharfen Abbildung. *Spektrum der Wissenschaft*, **7**:12–13.

[19] Scheuermann, T., (1997). *Berührungslose Gestaltvermessung von Mikrostrukturen durch Fokussuche*. Dissertation, Universität Stuttgart, Pfinztal: Fraunhofer ICT.

[20] Hempel, E., (1996). *Entwicklung eines Fuzzy-Systems zur qualitativen Bewertung von Grauwertbildern*. Master's thesis, Universität Karlsruhe.

[21] Pfundt, G., (1996). *Entwicklung eines Verfahrens zur konfokalen Mikrotopographievermessung mit Strukturprojektion*. Master's thesis, Universität Karlsruhe.

[22] Wagner, K., (1996). *Regelungsverfahren zur optimalen Ausleuchtung digitalisierter Grauwertbilder*. Master's thesis, Fachhochschule Karlsruhe.

[23] Graf, M., (1997). *Schichtdickenmessung durch Strukturprojektion und Fokussuche*. Master's thesis, Universität Karlsruhe.

[24] Beyer, H. and Riesenberg, H., (1988). *Handbuch der Mikroskopie*. Berlin: VEB Verlag Technik.

[25] Stokseth, P. A., (1969). Properties of a defocused optical system. *Jour. Opt. Soc. Am.*, **59**(10):1314–1321.

[26] Wilson, T. and Carlini, A. R., (1987). Size of the detector in confocal imaging systems. *Opt. Letters*, **12**(4):227–229.

[27] Hopkins, H. H., (1950). *Wave Theory of Aberrations*. Oxford: Clarendon Press.

[28] Hopkins, H. H., (1955). The frequency response of a defocused optical system. *Proc. R. Soc. London*, **231**:91–103.

[29] Wilson, T. and Carlini, A. R., (1989). The effect of aberrations on the axial response of confocal imaging-systems. *Jour. Microsc.*, **154**:243–256.

19 Processing of Digital Elevation Maps

Pierre Soille

Silsoe Research Institute, Silsoe, Bedfordshire, United Kingdom

19.1 Introduction

Digital elevation maps or models (DEMs) are arrays of numbers representing the spatial distribution of terrain elevations. They can be seen as gray-scale images, whereby the value of a pixel represents an elevation rather than a luminance intensity (the brighter the gray-tone level of a pixel, the higher the elevation of the terrain point corresponding to this pixel). Useful applications of DEMs can be found in civil/rural engineering, geographic information systems (GIS), geomorphology, water resources management, photogrammetry, satellite imaging, etc. A comprehensive review of hydrological, geomorphological, and biological applications of DEMs is proposed in [1].

A low-cost solution for generating a DEM consists in interpolating the elevation values between the elevation contour lines extracted from digitized topographic maps. For example, a sample of a digitized topographic map is shown in Fig. 19.1a. The contour lines of this image

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

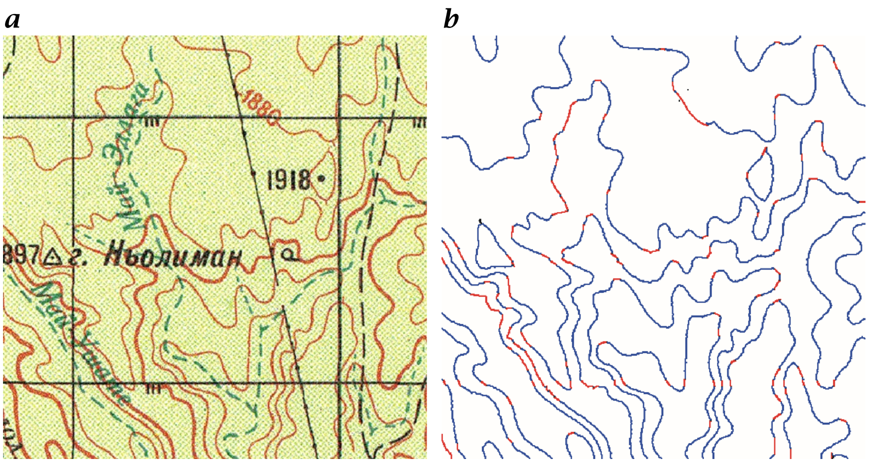


Figure 19.1: ***a*** *Digitized topographic map;* ***b*** *extracted contour lines (connections of initially disconnected lines are shown in red). From [2]; (see also Plate 2).*

can be automatically extracted using color information. The resulting image is then cleaned using size criteria (surface area opening and closing). The filtered contour lines are then thinned using a skeletonization algorithm. Finally, disconnected lines are reconnected by considering the distance separating their extremities as well as their orientation. The resulting contour lines are shown in Fig. 19.1b.

Once the contour lines have been assigned their elevation value, it is still necessary to interpolate the elevation values of the points located in between two successive contour lines. The first application detailed in this chapter (Section 19.2) concerns an interpolation technique based on geodesic distance and morphological reconstruction transforms.

The extraction of drainage networks from DEMs is essential for simulating river flows in response to a given rainfall. A technique based on the computation of drainage surface area is described in Section 19.3. Finally, we show in Section 19.4 that the automatic delineation of catchment basins on DEMs can be achieved with the watershed transformation detailed in Volume 2, Section 21.5.4.

19.2 Geodesic interpolation of contour data

A larger image of elevation contour lines is shown in Fig. 19.2: each line has a constant gray-scale value corresponding to the terrain elevation along the line and the elevation values of the pixels belonging to white regions in between two successive contour lines are unknown.

In this example, contour lines are associated with terrain elevation values but they could represent other spatial-dependent characteristics

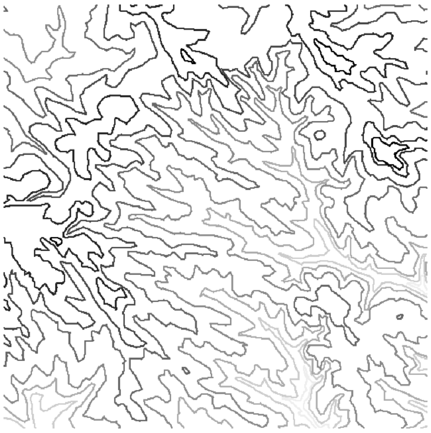


Figure 19.2: *A larger sample of an image CL of elevation contour lines. Elevation values are constant along the lines and only known along these lines.*

of an object. For instance, one could transform a gray-tone image into an image of iso-intensity contour lines and then attempt to regenerate the original image (i.e., lossy image compression technique). There is, therefore, a need for interpolating values located in between successive contour lines. Geodesic transformations are at the basis of an efficient method taking advantage of the topological and morphological properties of the contour lines. The method requires two steps [3]: i) generation of two plateau images; and ii) interpolation along the steepest slope lines.

19.2.1 Plateau image generation

A magnified section of an image of contour lines is shown in Fig. 19.3a. The pixel p lies within a white region surrounded by two contour lines referred to as its lower and upper contour lines. The lower contour line of a pixel p is denoted by $C_l(p)$ and the upper one by $C_u(p)$. These two contour lines may be identical as shown in Fig. 19.3b or when p is located precisely on a contour line.

The elevation of both the lower and upper contour lines associated with each pixel is determined by calculating two plateau images P_l and P_u from the image CL of contour lines:

$$\begin{aligned} P_l(p) &= CL[C_l(p)] \\ P_u(p) &= CL[C_u(p)] \end{aligned}$$

Hereafter, we assume that the image CL of contour lines has values strictly greater than 0 along the contour lines, the remaining pixels being set to 0 (i.e., unknown values to interpolate). The computation of

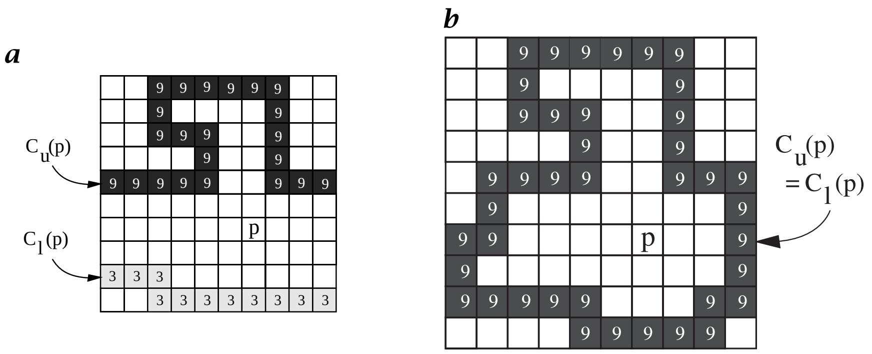


Figure 19.3: *Magnified section of an image of iso-intensity contour lines with a pixel p located in a white region where intensity values are unknown:* ***a*** *p lies between two successive contour lines;* ***b*** *p lies inside a closed contour line. $C_l(p)$ and $C_u(p)$ are, respectively, the lower and upper contour lines of the pixel p. Note that these contour lines are identical in* ***b****.*

the plateau images requires the definition of the following image M:

$$M(p) = \begin{cases} CL(p) & \text{if } p \text{ belongs to a contour line} \\ t_{\max} & \text{otherwise} \end{cases} \tag{19.1}$$

The plateau images are then obtained by performing a morphological reconstruction by erosion of CL from M for the lower plateau image P_l and a morphological reconstruction by dilation of M from CL for the upper plateau image P_u:

$$P_l = R^{\star}_{CL}(M) \tag{19.2}$$

$$P_u = R_M(CL) \tag{19.3}$$

The procedure is illustrated in Fig. 19.4 on 1-D profiles. Figure 19.4a is a 1-D profile of an image of contour lines. The image M is generated according to Eq. (19.1) and is shown in Fig. 19.4b. The resulting plateau images are shown in Fig. 19.4c and d. P_l and P_u give for each pixel the elevation of, respectively, the lower and upper contour lines.

The methodology applied to the image of contour lines shown in Fig. 19.2 outputs the plateau images shown in Fig. 19.5.

19.2.2 Interpolation along steepest slope lines

In Section 19.2.1, morphological reconstruction algorithms have been used for generating a pair of plateau images starting from an image of elevation contour lines. This was the first step towards the interpolation of the elevation values of the pixels located in between two successive contour lines. The second step consists in calculating the

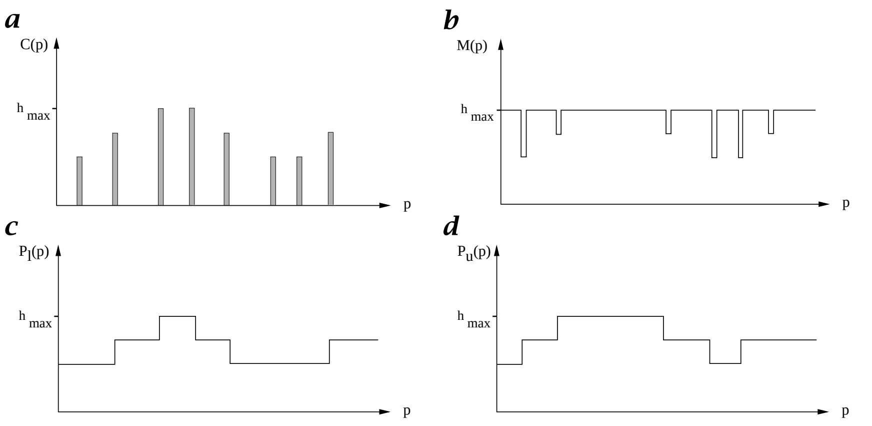


***Figure 19.4:** Plateau generation in the 1-D case. Starting from a contour line representation in **a**, plateau images are automatically generated using morphological reconstruction with appropriate pairs of mask/marker images: **b** image M derived from CL using Eq. (19.1); **c** plateau image $P_l = R^{\star}_{CL}(M)$; **d** plateau image $P_u = R_M(CL)$.*

value of each pixel p by a linear interpolation along the geodesic going from its upper contour line $C_u(p)$ to its lower contour line $C_l(p)$ and passing through p.

A magnified section of an image of contour lines is shown in Fig. 19.6. In this figure, a pixel p located in between its upper and lower contour lines is drawn with the corresponding geodesic linking the pixel to both contour lines. The shortest path going from $C_u(p)$ to $C_l(p)$, included in M and passing through p, is made of the geodesics from p to $C_l(p)$ and from p to $C_u(p)$.

We assume that the difference of elevation between $C_u(p)$ and $C_l(p)$ is evenly distributed along the geodesic path. The interpolated value $H(p)$ of p equals, therefore, the weighted mean of $P_u(p)$ and $P_l(p)$ by the geodesic distances from p to $C_u(p)$ and to $C_l(p)$, respectively:

$$H(p) = \frac{P_l(p)\ d_M[p, C_u(p)] + P_u(p)\ d_M[p, C_l(p)]}{d_M[p, C_l(p)] + d_M[p, C_u(p)]} \tag{19.4}$$

where the geodesic mask M is the set difference between the image definition domain and the contour lines, and P_u and P_l are the plateau images defined in Section 19.2.1. The length of the geodesics linking each pixel to its upper and lower contour lines is determined from two geodesic distance functions: from odd to even contour lines and vice versa (odd contour lines are obtained by considering one contour line out of two and starting from the contour line at the lowest level). The two marker sets R_1 and R_2 are defined as the set of even and odd contour lines, respectively. The two corresponding geodesic masks M_1

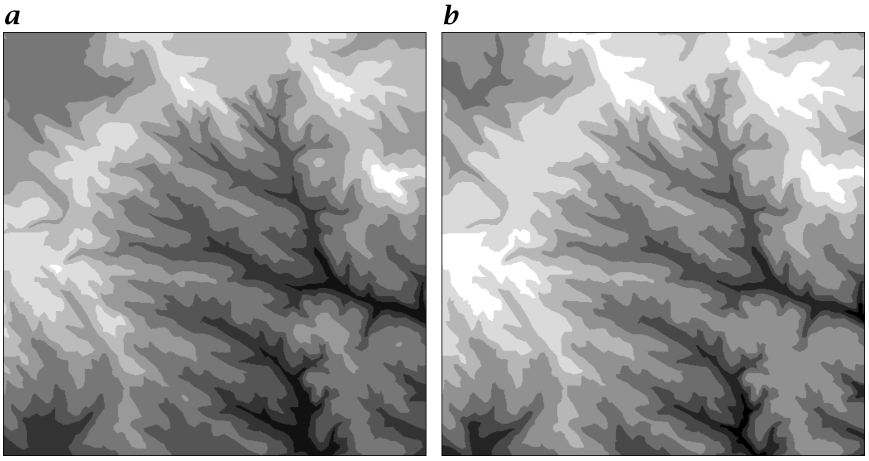


Figure 19.5: *Plateau images derived from Fig. 19.2 and using **a** Eq. (19.2) and **b** Eq. (19.3).*

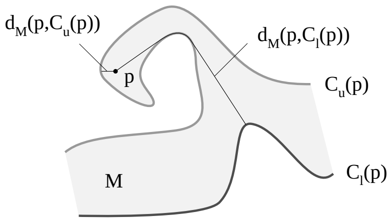


Figure 19.6: *Interpolation of the value of a point p located in between its lower C_l and upper C_u iso-intensity contour lines. The geodesic mask M is defined as the space delineated by these two contour lines.*

and M_2 are defined as the image domain of definition except the set of odd contour lines for M_1 and even contour lines for M_2. The geodesic masks, reference images, and the geodesic distance functions derived from Fig. 19.2 are provided in Fig. 19.7.

The geodesic distance functions $D_{M_1}(R_1)$ and $D_{M_2}(R_2)$ shown in Fig. 19.7e and f allow us to calculate the geodesic distances required by Eq. (19.4). The resulting interpolation on the image of contour lines is shown in Fig. 19.8.

The geodesic interpolation works for peak and pit configurations where a contour line is reduced to a unique point. This is illustrated in Fig. 19.9 for the interpolation of the elevation values of the points surrounded by the very last elevation contour line of a hill.

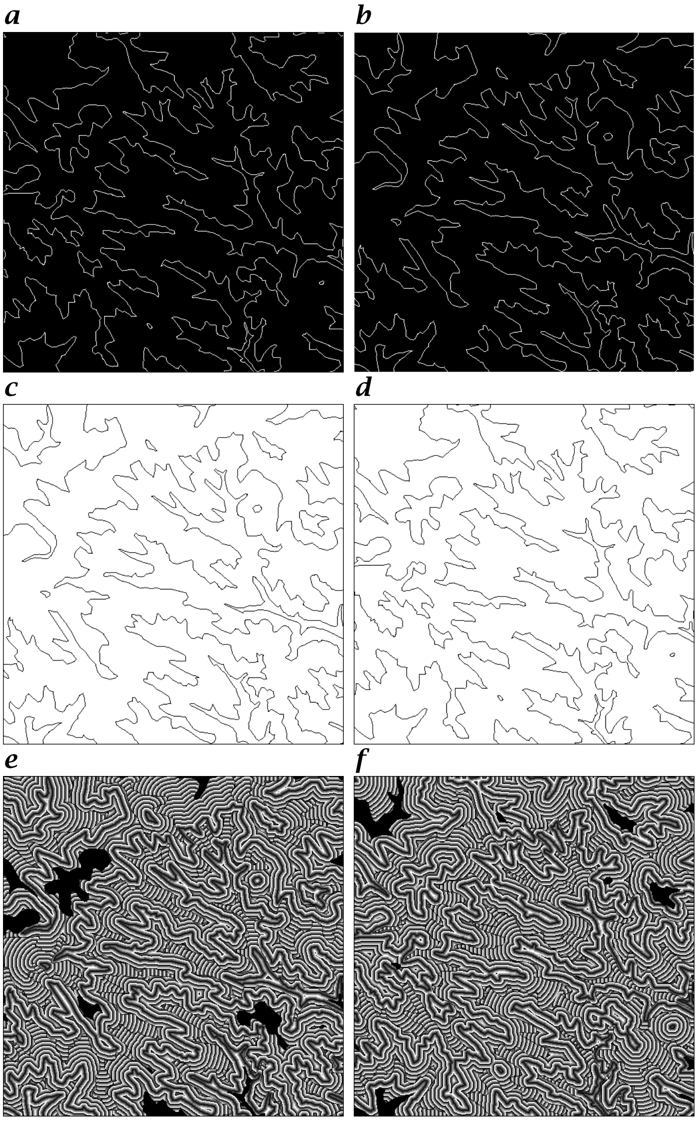


***Figure 19.7:** Geodesic distance functions from even to odd contour lines (**a**, **c** ,**e**) and vice versa (**b**, **d**, **f**): **a** geodesic mask M_1, that is, whole image plane except odd contour lines; **b** geodesic mask M_2, that is, whole image plane except even contour lines; **c** marker set R_1, that is, even contour lines; **d** marker set R_2, that is, odd contour lines; **e** geodesic distances $D_{M_1}(R_1)$; **f** geodesic distances $D_{M_2}(R_2)$. Odd and even contour lines are extracted from the iso-intensity contour lines of Fig. 19.2.*

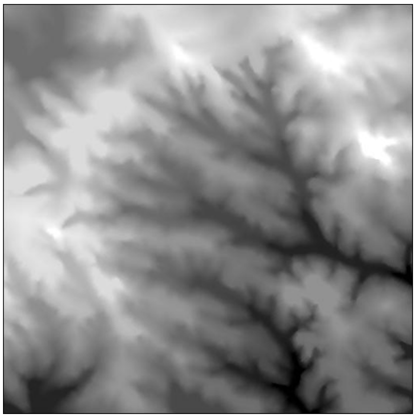


Figure 19.8: *Interpolated image from the image of contour lines of Fig. 19.2 and using Eq. (19.4).*

19.2.3 Extensions

The geodesic interpolation technique presented here can be directly extended to the interpolation of intermediate cross-sections of 3-D binary images [4]. Indeed, it provides us with a model for deforming a given shape into another one. Applications to the interpolation of temporal sequences of images for image compression and coding have also been considered in Salembier et al. [5] and Marcotegui and Meyer [6]. Finally, the method has been adapted for the interpolation of mosaic images in Meyer [7]. Pseudo-code for Euclidean geodesic distance functions is given in [3].

19.3 Drainage network detection

Drainage networks are topographic features where water run-off is concentrated. They represent a fundamental concept in earth sciences, and their extraction from DEMs is at the basis of most quantitative studies in hydrology and water management.

Two distinct approaches may be considered for defining drainage networks. The first one is a morphological approach where pixels belonging to the drainage network are defined from local morphologies, for example, [8, 9, 10, 11, 12]. The most common morphology considered is a curvature coefficient. For example, all pixels having a concave curvature coefficient higher than a given threshold are considered drainage pixels. This approach has two major drawbacks: i) there exists no absolute threshold level to discriminate drainage pixels from other pixels and ii) the drainage networks obtained are generally dis-

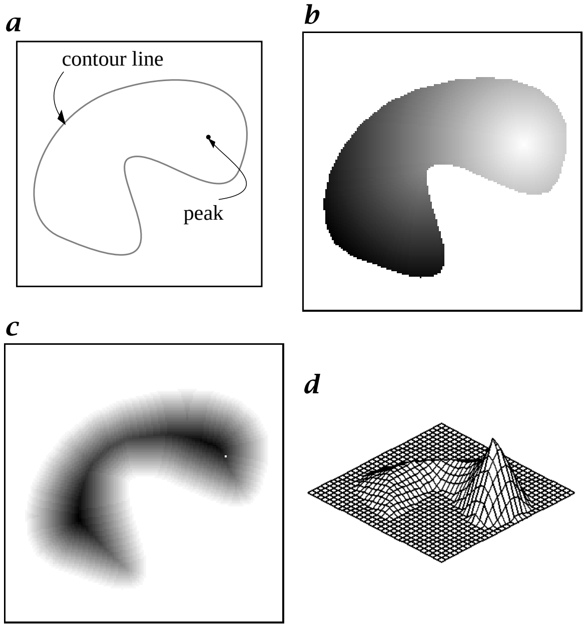


Figure 19.9: *The interpolation technique based on geodesic distances is suited to the interpolation of summits:* ***a*** *input image, elevation values are known only along the contour line and at the highest point surrounded by the contour line;* ***b*** *geodesic distance function from the peak;* ***c*** *geodesic distance function from the contour line;* ***d*** *resulting interpolation.*

connected because the pixels are processed separately. Additional processing steps are therefore needed to connect extracted pixels and to remove irrelevant ones. The second approach is a hydrological approach where a flow of water is simulated over the topographic surface. The downstream DW of a pixel p is defined as the steepest slope path starting from p. The contributing drainage area CDA of a pixel p is then defined as the number of pixels whose downstream goes through p:

$$CDA(p) = \#\{p' \mid p \in DW(p')\} \tag{19.5}$$

Drainage pixels are then defined as the pixels having a contributing drainage area greater than a given threshold S:

$$p, \text{ drainage pixel } \Leftrightarrow CDA(p) \geq S$$

This definition was first proposed by Speight [13]. Contrary to the morphological approach, the resulting networks are always connected. However, difficulties often arise when determining the steepest slope paths because downstreams may be interrupted by meaningless minima, and flow directions through plateaus are not defined [14]. A plateau at level h is an equivalence class of pixels defined by the equivalence relationship $\mathcal{R}$, such that

$$p \mathcal{R} q \Leftrightarrow \text{there exists an iso-intensity path linking } p \text{ to } q$$

The descending border of a plateau is the set of pixels of the plateau having at least one neighbor at a lower altitude than that of the plateau. Flow directions remain undefined for all other plateau pixels.

The algorithm presented in [15] and summarized hereafter implements the hydrological definition of drainage networks. It relies on a morphological preprocessing of the DEMs aiming at removing all irrelevant minima and plateau regions.

19.3.1 Removal of irrelevant minima

The fluvial erosion processes will usually not produce minima at the current spatial resolution of DEMs [16]. We therefore assume that all minima of a DEM that are not connected to the image border are artifacts or data errors (except for special terrains such as volcanic, karst, or recently glaciated landscapes). Algorithms based on local filtering operations such as convolutions or morphological closings usually fail to remove all irrelevant minima and may even produce new minima. Soille and Ansoult [17] showed that a morphological reconstruction by erosion can be used to solve this problem. The filter consists of a generalization of the transformation used for removing the holes of the connected components of a binary image. Indeed, the holes of a binary image are defined as the set of its regional minima not connected to the image border. This definition applies directly to gray-scale images and can be implemented using a minima imposition technique. The marker image f_m used in the morphological reconstruction by erosion is set to the maximum image value except along its border where the values of the original image are kept:

$$\text{FILL}(f) = R^{\star}_{f_m}(f)$$

where

$$f_m(x) = \begin{cases} f(x) & \text{if } p \text{ lies on the border of } f \\ t_{\max} & \text{otherwise} \end{cases}$$

This morphological reconstruction corresponds to a generalization of the algorithm used to fill holes of binary objects. It is illustrated in Fig. 19.10 for a 1-D signal (see Fig. 19.17c for an example on a 2-D image).

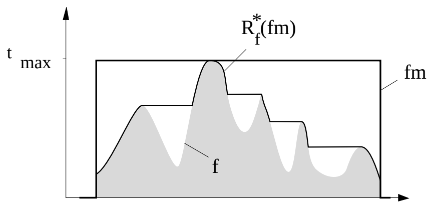


Figure 19.10: *Fillhole on a 1-D signal f: all inner minima of f are removed by the morphological reconstruction of f from the marker function f_m.*

a

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

b

	0	0					
	0	1	2	3	4		
			3	4	5		
		5	4	5	6		
		6	5				

c

1	1	1	1	1	1	1	1
1	2	2	9	9	9	9	1
1	2	3	4	5	6	9	1
1	9	9	5	6	7	9	1
1	9	7	6	7	8	9	1
1	9	8	7	10	10	9	1
1	9	9	9	9	9	9	1
1	1	1	1	1	1	1	1

Figure 19.11: *Lower complete transformation:* ***a*** *original image, plateau in gray;* ***b*** *geodesic distance function on the plateau from the descending border pixels (in gray), using 4-connectivity; and* ***c*** *lower complete image.*

19.3.2 Determination of flow directions

The drainage direction at a point p is the direction of the neighbor of p that has the lowest elevation and has a lower elevation than p. If several neighbors satisfy these conditions one is chosen arbitrarily among them. Within plateaus, no neighbor pixel satisfies these conditions and drainage directions remain undefined. A solution may consist in looking for a point of lower elevation in a larger neighborhood. A better approach is to transform the input DEM into a *lower complete* image. A lower complete image [18] is an image whose only points having no lower neighbor are the points of the regional minima. The lower complete image is constructed by adding an auxiliary relief to the plateaus. This auxiliary relief corresponds to the geodesic distance function of the descending borders of the plateau inside the plateau. Thus, the value of the auxiliary relief at a given plateau pixel p corresponds to the length of the geodesic (i.e., the shortest path) linking p to the descending border of the plateau. This is illustrated in Fig. 19.11 with numerical values.

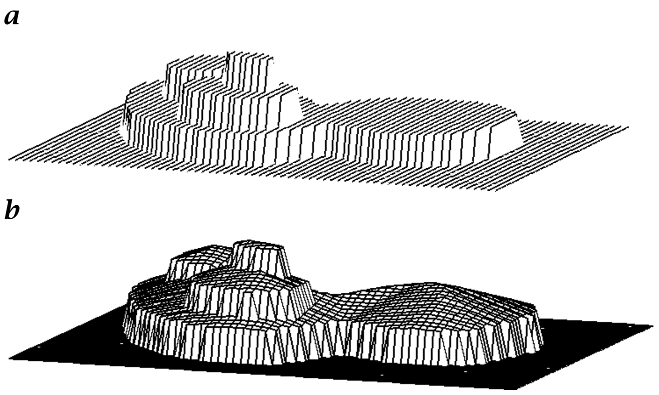


Figure 19.12: ***a*** *Perspective view of a plateau image, and* ***b*** *of its lower complete version.*

Figure 19.12 illustrates the lower complete transformation with a perspective view of plateaus (Fig. 19.12a), and its lower complete version (Fig. 19.12b).

Note that values of points higher than the plateau are increased by the largest computed geodesic distance (Fig. 19.11c). Consequently, the resulting DEM has more levels then the original one, but ordered relationships between pixels are preserved (except on plateaus, by definition).

The lower complete transformation can be achieved by means of sequential algorithms as described in [19]. A more efficient method based on hierarchical queues of pixels is detailed in [15]. The lower complete transformation of the DEM allows the determination of the flow direction for each point of the model because the only pixels of a lower complete image having no lower neighbor are the points of the regional minima. These minima correspond to the outlets of the drainage networks. Figure 19.13b shows the image of flow directions of the DEM presented in Fig. 19.13a.

19.3.3 Flow simulation

We now simulate a flow of water for determining the contributive drainage area CDA of each pixel. According to Eq. (19.5) the CDA of a pixel equals its surface area (i.e., 1 if it is computed in number of pixels) plus the sum of the CDA of its neighbors draining through it:

$$CDA(p) = 1 + \sum_{i \mid p_i \in N_G(p) \wedge p \in DW(p_i)} CDA(p_i)$$

The image of cumulative drainage areas is therefore initialized to 1. We then start to simulate the flow from the pixels at the highest elevation

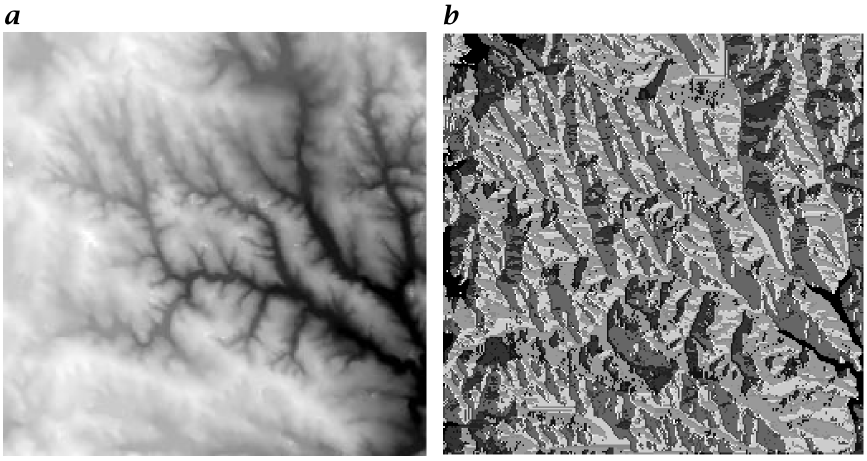


Figure 19.13: ***a*** *DEM and* ***b*** *the corresponding image of flow directions.*

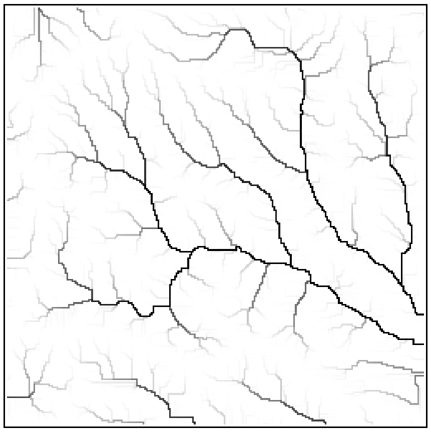


Figure 19.14: *Image of contributing drainage areas of the DEM presented in Fig. 19.13a.*

and let them drain towards their lowest neighbor. When a pixel drains towards another, the CDA of the latter is incremented by that of the former. The process is then repeated for the second highest elevation and so forth until the lowest elevation is reached. When a pixel has more than one neighbor at the lowest elevation, one of them is chosen randomly to avoid systematic errors. The resulting image of contributing drainage areas is shown in Fig. 19.14. This image can be thresholded at a user-defined value S to extract the drainage network pixels draining at least S pixels. Figure 19.15 shows the drainage networks for various threshold levels.

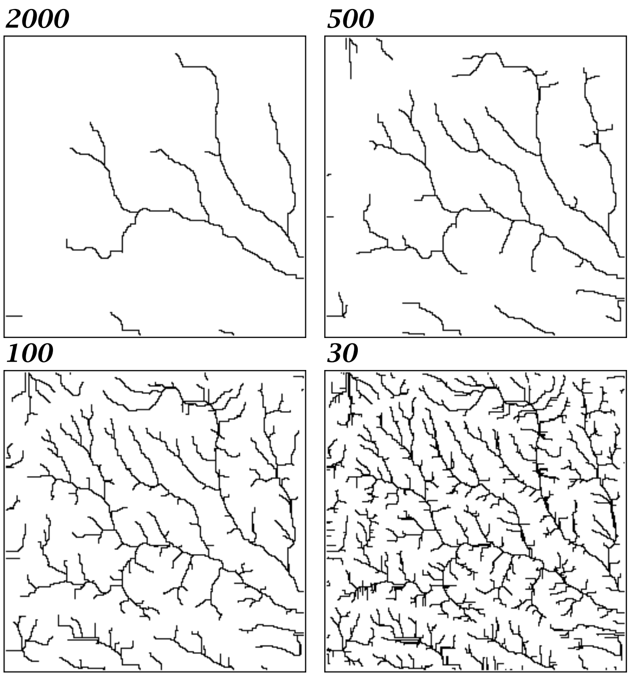


Figure 19.15: *Thresholdings of the image of contributing drainage areas shown in Fig. 19.14. Threshold values are in number of pixels.*

Notice that DEMs are also used in bathymetry [20]. The detection of undersea canyons along the continental margins can be achieved with the presented methodology. For example, Fig. 19.16 represents the undersea 'drainage networks' or canyons superimposed on the input DEM.

19.4 Watershed detection

Due to its very nature, the *watershed transformation* (Volume 2, Section 21.5.4) can be applied to digital elevation models (DEMs) for delineating their catchment basins. However, a direct computation of the watersheds of a DEM leads to an over-segmented DEM. For example, Fig. 19.17a is a DEM of a mountainous area with a dense hydrographic network and Fig. 19.17b represents the watersheds of this DEM. There is an over-segmentation because all catchment basins should be connected to the border of the DEM. Indeed, topographic surfaces exposed to fluvial erosion processes do not normally contain any regional minima at the current spatial resolution of DEMs: the water flows down until it reaches the sea. It follows that all minima not connected to the

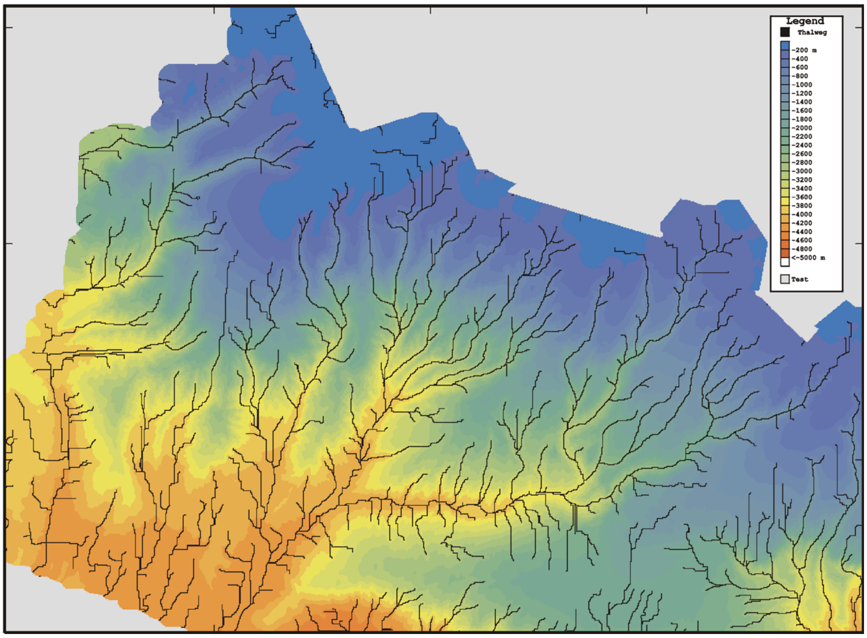


Figure 19.16: *Submarine drainage networks extracted from bathymetric data of the approaches of the Channel in the NE Atlantic. The cell size is* $200\,m \times 200\,m$*. The network has been extracted using a threshold of 300 cells for the cumulative drainage area. The canyon network incises the continental slope from the shelf (-200 m) to the abyssal plain (-4500 m); (see also Plate 3).*

image border must be removed before applying the watershed transformation. This is achieved with the morphological reconstruction by erosion detailed in Section 19.3.1. The internal regional minima of the input DEM are displayed in Fig. 19.17c. The watersheds of the 'filled' DEM produce the correct segmentation [17]. They are superimposed on the original DEM in Fig. 19.17d.

19.5 Concluding remarks

The subgraph representation of a gray-scale image used in morphological image processing corresponds to a topographic representation. It is therefore not surprising that morphological operators are well-suited to the processing of DEMs. In fact, many morphological operators such as the watershed transformation are based on geomorphological concepts. The techniques presented in this chapter have been illustrated on regular DEMs defined over a square grid. They all apply to the less common triangulated irregular networks (TINs) whereby the topographic surface is sampled on surface specific points such as peaks,

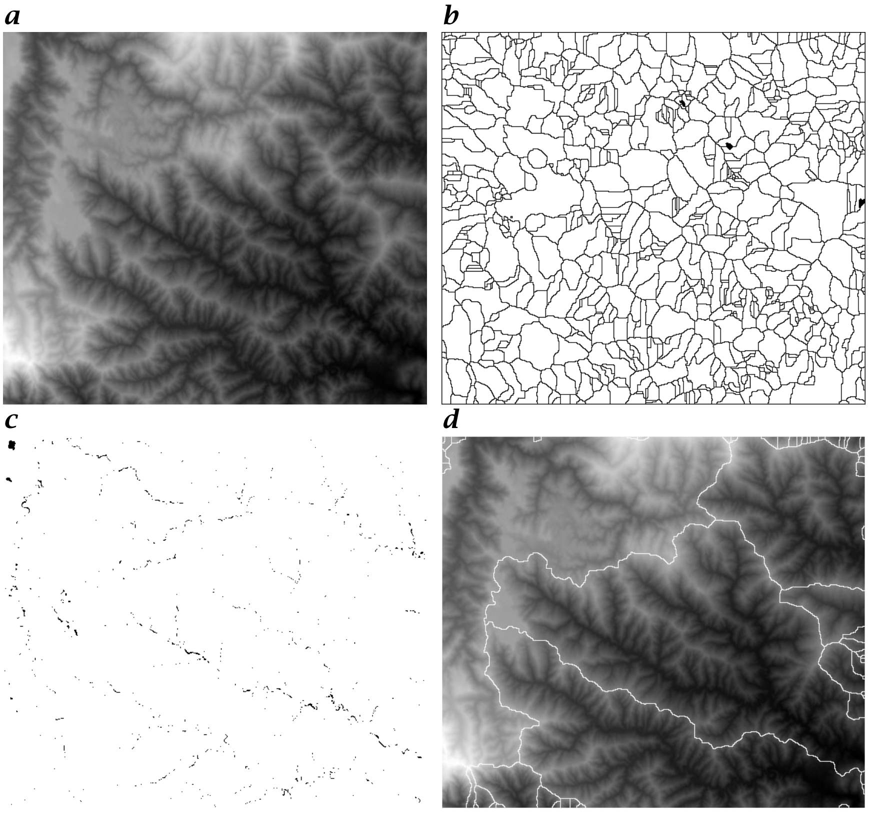


Figure 19.17: *Automatic delineation of the catchment basins of a digital elevation model (DEM) using the watershed transformation:* ***a*** *input digital elevation model (DEM);* ***b*** *watershed on original DEM;* ***c*** *internal regional minima of the input DEM, that is, $FILL(a) - a$.* ***d*** *watershed of 'filled' DEM superimposed on input DEM. The fillhole transform must be applied to the input DEM before applying the watershed transformation so as to avoid oversegmentation.*

ridges, and breaks in slope. Indeed, the neighboring relationships between the points define a graph that can be processed with morphological operators (see [21] for a general introduction about graph morphology and [22] for watersheds on graphs).

19.6 References

[1] Moore, I., Grayson, R., and Ladson, A., (1991). Digital terrain modeling: a review of hydrological, geomorphological, and biological applications. *Hydrological Processes*, **5**:3–30.

[2] Arrighi, P. and Soille, P., (1999). From scanned topographic maps to digital elevation models. In *Proc. of Geovision'99*, E. Pirard, ed. Université de Liège. To appear.

[3] Soille, P., (1991). Spatial distributions from contour lines: an efficient methodology based on distance transformations. *Jour. Visual Communication and Image Representation*, **2**(2):138–150.

[4] Usson, Y. and Montanvert, A., (1993). Reconstruction tridimensionnelle à partir des distances discrètes. In *Colloque géométrie discrète en imagerie : fondements et applications*, pp. 20–21. Strasbourg.

[5] Salembier, P., Brigger, P., Casas, J., and Pardàs, M., (1996). Morphological operators for image and video compression. *IEEE Trans. Image Processing*, **5**(6):881–898.

[6] Marcotegui, B. and Meyer, F., (1994). Morphological segmentation of image sequences. In *Mathematical Morphology and Its Applications to Image Processing*, J. Serra and P. Soille, eds., pp. 101–108. Dordrecht: Kluwer Academic Publishers.

[7] Meyer, F., (1996). A morphological interpolation method for mosaic images. In *Mathematical Morphology and Its Applications to Image and Signal Processing*, P. Maragos, R. Schafer, and M. Butt, eds., pp. 337–344. Boston: Kluwer Academic Publishers.

[8] Haralick, R., (1983). Ridges and valleys on digital images. *Computer Vision, Graphics, and Image Processing*, **22**:28–38.

[9] Seemuller, W., (1989). The extraction of ordered vector drainage networks from elevation data. *Computer Vision, Graphics, and Image Processing*, **47**:45–58.

[10] Toriwaki, J. and Fukumara, T., (1978). Extraction of structural information from grey pictures. *Computer Graphics and Image Processing*, **7**:30–51.

[11] Johnston, E. and Rosenfeld, A., (1975). Digital detection of pits, peaks, ridges, and ravines. *IEEE Trans. Systems, Man and Cybernetics*, **5**:472–480.

[12] Zhang, M., Campbell, J., and Haralick, R., (1990). Automatic delineation of drainage basins within digital elevation data using the topographic primal sketch. *Mathematical Geology*, **22**(2):189–209.

[13] Speight, J., (1968). Parametric description of land form. In *Land Evaluation*, G. Stewart, ed., pp. 239–250. CSIRO/UNESCO.

[14] O'Callaghan, J. and Mark, D., (1984). The extraction of drainage networks from digital elevation data. *Computer Vision, Graphics, and Image Processing*, **28**:323–344.

[15] Soille, P. and Gratin, C., (1994). An efficient algorithm for drainage networks extraction on DEMs. *Jour. Visual Communication and Image Representation*, **5**(2):181–189.

[16] Band, L., (1986). Topographic partition of watersheds with Digital Elevation Models. *Water Resources Research*, **22**(1):15–24.

[17] Soille, P. and Ansoult, M., (1990). Automated basin delineation from digital elevation models using mathematical morphology. *Signal Processing*, **20**:171–182.

[18] Meyer, F. and Beucher, S., (1990). Morphological segmentation. *Jour. Visual Communication and Image Representation*, **1**(1):21–46.

[19] Meyer, F., (1989). Skeletons and perceptual graphs. *Signal Processing*, **16**: 335–363.

[20] Bourillet, J.-F., Edy, C., Rambert, F., Satra, C., and Loubrieu, B., (1996). Swath mapping system processing: bathymetry and cartography. *Marine Geophysical Researches*, **18**:487–506.

[21] Vincent, L., (1989). Graphs and mathematical morphology. *Signal Processing*, **16**:365–388.

[22] Vincent, L. and Soille, P., (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **13**(6):583–598.

20 Three-dimensional Modeling of Objects from Image Sequences

Reinhard Koch

Dept. Elektrotechniek, Katholieke Universiteit Leuven, Belgium

20.1 Introduction

The use of *3-D surface models* for the purpose of information processing is gaining importance. Highly realistic 3-D models are readily used to visualize and simulate events, for example in flight simulators, in games and the film industry or for product presentations. The range of applications spans from architecture visualization [1] over virtual

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

television studios [2], virtual presence for video communications [3] to general virtual reality applications (see Chapter 24).

A limitation to the widespread use of these techniques is currently the high cost of such 3-D models because they have to be produced manually. Especially, if existing objects are to be reconstructed, the measurement process for obtaining the correct geometric and photometric data is tedious and time consuming. With *image-based 3-D acquisition systems* we intend to solve this problem. Such systems should ideally require only a set of images or a video of the object from various viewpoints. The system should then automatically extract a complete 3-D surface description of the observed object.

3-D scene measurements can be obtained with either active or passive techniques. With active techniques, some kind of energy is projected onto the scene and the energy response is measured. This class of range sensors is described in detail elsewhere in this handbook (see Volume 1, Chapters 18–20, Chapters 17 and 21). They usually give the highest measurement precision, but at the cost of expensive dedicated acquisition hardware and a principally limited measurement range. The passive techniques, on the other hand, rely only on the natural image content as given by the camera. Because of this, they can not guarantee as precise measurements as with the active techniques. The great advantage of passive techniques, however, is their simple acquisition hardware (one or two cameras suffice) and the wide depth range that is limited only by the displacement of the cameras. With a single-camera structure from motion approach as described in this chapter it is possible to model complete landscapes as well as small objects or buildings.

Passive 3-D modeling from image sequences is still a recent topic in computer vision research. Great effort went into developing algorithms that estimate 3-D object shape from various sources, termed *shape from motion*, *stereo*, and others [4, 5]; see also Volume 1, Chapter 18. On the other hand, research was conducted to find solutions to the problem of rigid object motion [6, 7]. The problem of dynamic nonrigid bodies was addressed by Terzopoulos and Metaxas [8] and Pentland and Horowitz [9]. Some approaches are reported from monocular vision systems to compute dense depth maps for orthographic [10] and perspective projection [11, 12] and from calibrated stereoscopic systems [13, 14, 15, 16].

Metric reconstruction from uncalibrated sequences is still under investigation. In the uncalibrated case all parameters—camera pose and intrinsic calibration as well as the 3-D scene structure—have to be estimated from the 2-D image sequence alone. Faugeras [17] and Hartley [18] first demonstrated how to obtain projective reconstructions from such image sequences. Since then, researchers tried to find ways to upgrade these reconstructions to metric (i. e., Euclidean but unknown scale, see, for example, Heyden and Åström [19], Pollefeys and Van-

Gool [20], Pollefeys et al. [21], Triggs [22]). While most of the research focused on certain aspects of the modeling such as camera tracking, calibration, or depth estimation, only few systems were developed that cover all aspects of 3-D modeling from camera tracking and calibration over depth estimation to the generation of textured 3-D models.

The application described here discusses such a complete *3-D modeling system*. It automatically models rigid 3-D objects that are observed by a moving camera or that rotate in front of the camera. Geometry and surface texture as well as object and camera motion are extracted from the image sequence alone. Section 20.2 sketches the general structure of the modeling system. Section 20.3 discusses the camera self-calibration for uncalibrated monocular image sequences. In Section 20.4 the depth estimation from a single image pair is developed in some detail. Section 20.5 deals with the 3-D model generation. We will conclude with some examples.

20.2 System overview

The complete modeling system can be divided into the three main modules, *calibration*, *depth estimation*, and *3-D model building*.

Camera calibration. If depth has to be estimated from an image sequence a calibration of camera view points and intrinsic parameters is necessary. Depending on the available camera configuration different approaches can be used to calibrate the image sequence. The described system was originally designed to work with a *calibrated stereo rig* where the calibration is available from *a priori* off-line calibration [23]. The need to calibrate the rig and the specialized hardware necessary to record stereoscopic image sequences limit the practical use of this system. In order to deal with the more realistic situation where only a single camera is available and no calibration can be performed beforehand, we extended the system to work with monoscopic uncalibrated image sequences as well. Subsequent images of the sequence are then treated as stereoscopic image pairs thus allowing the use of the modeling system with a *virtual stereo rig*.

Stereoscopic depth estimation. Stereoscopic image pairs are used to extract depth maps of the scene. Different methods such as block matching, interpolation, and differential subpixel estimation are employed to achieve highest accuracy. Epipolar constraints as well as some scene geometry constraints are used to guide the correspondence search and to obtain a dense and accurate depth map.

Three-dimensional model building. The surface model generation itself is divided into three modules:

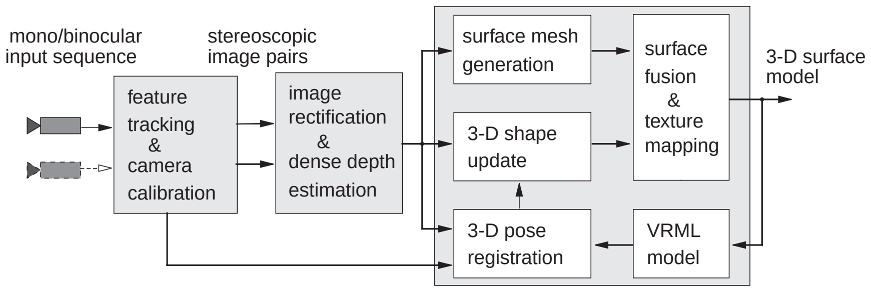

mono/binocular input sequence
stereoscopic image pairs
feature tracking & camera calibration
image rectification & dense depth estimation
surface mesh generation
3-D shape update
3-D pose registration
surface fusion & texture mapping
VRML model
3-D surface model

Figure 20.1: *Structure of the 3-D scene modeling system.*

- the depth measurements from each view point are converted into a 3-D surface model that is approximated by a triangular wire-frame;
- the different surfaces are registered in a consistent object coordinate system and the surface geometry is updated according to the depth measurements; and
- the different surfaces are fused together to form the complete object and the surface texture is mapped onto the model to add realism for visualization.

The resulting textured 3-D surface model is stored in VRML-format for easy interchange with visualization systems. The detailed structure of the complete modeling system is shown in Fig. 20.1.

20.3 Image acquisition and calibration

The imaging camera is described as a pinhole camera with perspective projection. In that case the camera projection matrix $\boldsymbol{P}$ has the form

$$\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}|\text{-}\boldsymbol{R}\boldsymbol{t}] \text{ with } \boldsymbol{K} = \begin{bmatrix} c_x & h_s & h_x \\ 0 & c_y & h_y \\ 0 & 0 & 1 \end{bmatrix} \tag{20.1}$$

where $\boldsymbol{R}$ indicates the camera rotation and $\boldsymbol{t}$ is the position of the projection center in world coordinates; $\boldsymbol{K}$ contains the internal camera parameters: c_x and c_y are the horizontal and vertical focal length (in pixels); $\boldsymbol{h} = [h_x, h_y]^T$ is the principal image point, and h_s is a measure of the image skew.

20.3.1 Off-line calibration

For the off-line calibration of the binocular stereo rig all camera parameters are calibrated with the calibration according to Tsai [24]. A planar, tilted calibration pattern is presented to the stereo rig from which the relative orientation and intrinsic parameters are estimated with high accuracy. The calibration computes a Euclidean metric with absolute scale in the calibration reference coordinate system. In fact, any well-established calibration technique can be used for this purpose (see also Volume 1, Chapter 17).

20.3.2 Online self-calibration

The approach of using a calibrated stereorig is feasible in controlled laboratory and indoor environments, but can not be applied to image acquisition in an unrestricted outdoor environment. Often it is not possible to carry along the heavy and delicate equipment that goes with stereoscopic recording devices. Another challenge is to reconstruct a scene from few images taken by an uncalibrated consumer photo camera without any chance to influence the image acquisition. These demands require additional efforts to analyze the image sequences and to calibrate them on the fly through self-calibration methods.

If uncalibrated image sequences are processed the system has to calibrate the camera orientation parameters $\boldsymbol{R}, \boldsymbol{t}$ and the intrinsic camera parameters $\boldsymbol{K}$ from the image data. This is achieved in a two-step approach. First, a projective calibration of the camera pose is obtained, followed by an estimation of the intrinsic parameters that remove the projective ambiguity based on some constraints.

Retrieving the projective framework. Our approach follows the procedure proposed by Beardsley et al. [25]. At first, feature correspondences are found by extracting intensity corners in images pairs and matching them using a robust corner matcher [26]. A random sampling method selects point samples in an *a priori* defined search region that are compared by intensity correlation. Different sample sets are then tested to find the *epipolar constraint* that is best supported by additional matches. The epipolar constraint states that the 3-D position of a point is restricted to the line passing through its image point and the camera projection center. Therefore the corresponding point is restricted to the projection of this line in the other image. Using this constraint, more matches can easily be found and used to refine this calibration. The principle is explained in Fig. 20.2. From the epipolar constraint a restricted calibration of the setup is calculated. This allows eliminating matches that are inconsistent with the calibration.

For each camera view point we obtain projection matrices (size 3×4) of the form

$$\boldsymbol{P}_1 = [\boldsymbol{I}|0] \text{ and } \boldsymbol{P}_i = [\boldsymbol{H}_{1i}|\boldsymbol{e}_{1i}] \tag{20.2}$$

where $\boldsymbol{H}_{1i}$ is the homography for some reference plane from view 1 to view i and $\boldsymbol{e}_{1i}$ is the corresponding epipole.

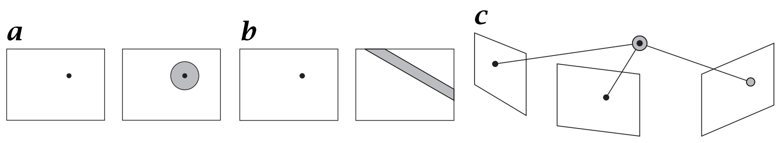


Figure 20.2: ***a*** *Image with a priori search region;* ***b*** *search region based on the epipolar constraint;* ***c*** *prediction of search region in the sequence after projective reconstruction of the point (used for refinement).*

Metric calibration. Such a projective calibration is certainly not satisfactory for the purpose of 3-D modeling. A reconstruction obtained up to a projective transformation can differ very much from the original scene according to human perception: orthogonality and parallelism are in general not preserved, part of the scene may be warped to infinity, etc. To obtain a metric calibration, constraints on the internal camera parameters can be imposed (e.g., absence of skew, known aspect ratio, ...). By exploiting these constraints, the projective reconstruction can be upgraded to metric structure where the projection matrices take the form of Eq. (20.1) [21].

Some results of point tracking and camera calibration are demonstrated with the *Castle* sequence. It consists of 22 images of 720×576 pixel resolution taken with a standard semiprofessional camcorder that was moved freely in front of the building. In Fig. 20.3 three of the images and the estimated 3-D-structure of the tracked points with calibrated camera positions are displayed. The tracked points are reconstructed in 3-D to give a sparse object structure. Comparison between the reconstruction and the real building geometry confirms the good metric qualities of the calibration (see [21]).

20.4 Stereoscopic depth estimation

In this section we will develop a dense correspondence matching algorithm based on the analysis of stereoscopic image pairs. For the algorithm it does not matter if we process binocular image pairs from a stereo rig or subsequent images from a moving camera because we assume a stationary scene. The camera intrinsic and relative extrinsic parameters were estimated in the previous (self-)calibration step.

20.4.1 Exploiting scene constraints

The epipolar constraint obtained from calibration restricts corresponding image points to lie in the epipolar plane that also cuts a 3-D profile out of the surface of the scene objects. The epipolar plane is the plane defined by the the image point and the camera projection centers (see

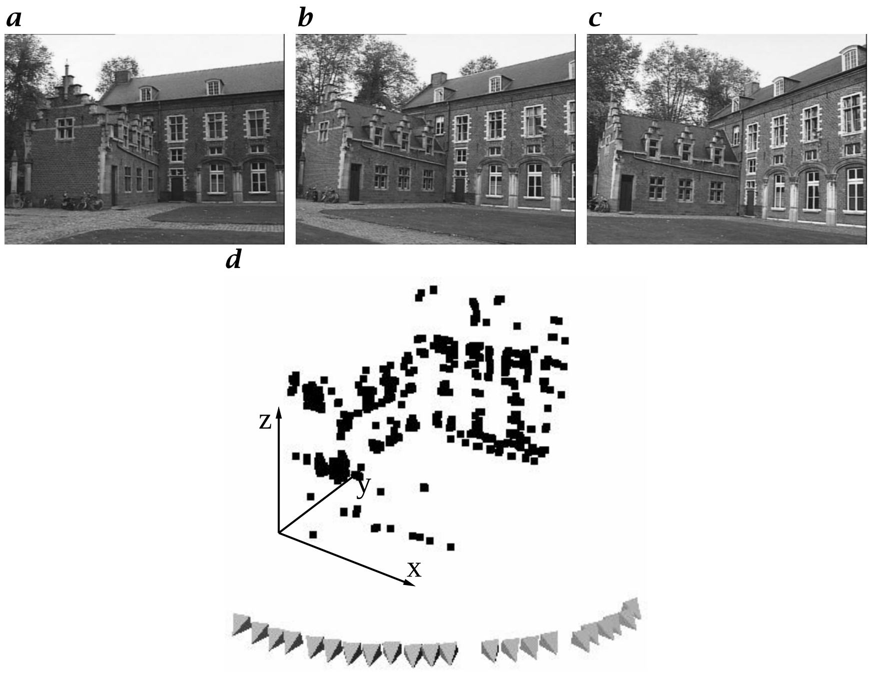


Figure 20.3: ***a*** *–* ***c*** *show 3 of 22 images of a castle;* ***d*** *a view of the reconstructed 3-D feature points in black and the calibrated camera positions visualized as pyramids.*

also Volume 2, Chapter 17). The profile projects onto the corresponding epipolar lines in the images l and r where it forms an ordered set of neighboring correspondences (see Fig. 20.4a).

For well-behaved surfaces this ordering is preserved and delivers an additional constraint, known as *ordering constraint*. Scene constraints such as this can be applied by making weak assumptions about the object geometry. In many real applications the observed objects will be opaque and composed from piecewise-continuous surfaces. If this restriction holds then additional constraints can be imposed on the correspondence estimation. Kochan [27] listed as many as 12 different constraints for correspondence estimation in stereo pairs. Of them, the two most important apart from the epipolar constraint are:

Ordering constraint. For opaque surfaces the order of neighboring correspondences on the epipolar line is always preserved. This ordering allows the construction of a dynamic programming scheme that is employed by many dense disparity estimation algorithms (see Volume 2, Chapter 18, Cox et al. [28], Falkenhagen [29], Gimel'farb [30]).

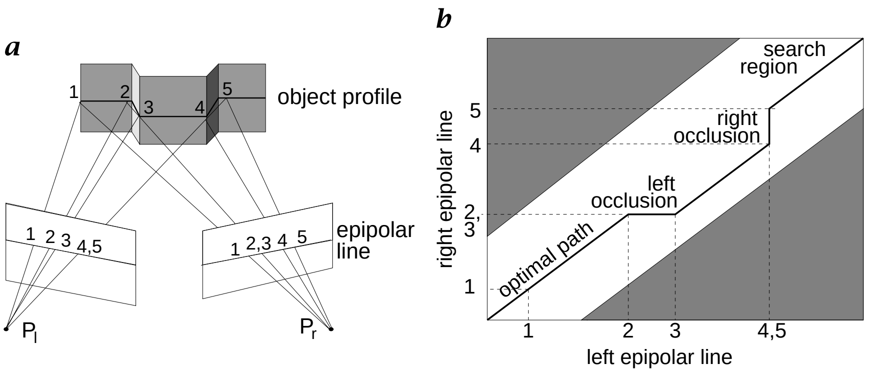


Figure 20.4: ***a*** *Object profile triangulation from ordered neighboring correspondences;* ***b*** *dynamic programming search path considering occlusions.*

Uniqueness constraint. The correspondence between any two corresponding points is bidirectional as long as there is no occlusion in one of the images. A correspondence vector pointing from an image point to its corresponding point in the other image always has a corresponding reverse vector pointing back. This constraint is used to detect outliers and occlusions.

20.4.2 Image rectification

All forementioned constraints operate along the epipolar line that may have an arbitrary orientation in the image planes. The matching procedure is greatly simplified if the image pair is rectified to standard geometry. In standard geometry both image planes are coplanar and the epipoles are projected to infinity. The rectified image planes can further be oriented such that the epipolar lines coincide with the image scan lines. The images are rectified by a planar projective mapping from the original towards the rectified image planes (Volume 2, Chapter 9). Conveniently the projective mapping is chosen such that the resulting image distortions are minimized.

20.4.3 Constrained block matching

For dense correspondence matching a disparity estimator based on the dynamic programming scheme of Cox et al. [28] is used that incorporates the forementioned constraints on rectified image pairs. At each pixel in the one image the matcher searches for maximum normalized cross correlation in the other by shifting a small measurement window (kernel size 5×5 or 7×7) along the corresponding scan line. The selected search step size (usually 1 pixel) determines the search resolution. Matching ambiguities are resolved by exploiting the ordering constraint for the complete scan-line simultaneously. The effect of occlusions on the scan-line search is sketched in Fig. 20.4b. The left and right scan-line span a 2-D search space that is bounded by a maximum allowable disparity range. Each entry in this space holds the corresponding cross correlation. Dynamic programming finds the disparity path with maximum cross correlation through the search space.

The basic algorithm as described in the foregoing was further adapted to employ extended neighborhood relationships and a pyramidal estimation scheme to deal reliably with very large disparity ranges of over 50 % of the image size [29].

20.4.4 Disparity interpolation

The disparity estimates computed with this approach give dense and reliable depth maps with a disparity resolution of 1 pixel due to quantization of the search space. This quantization leads to a discontinuous and rough surface that can not be tolerated in surface reconstruction. The data must be interpolated with a given smoothing function (Volume 2, Chapter 8). To solve this problem a disparity interpolation with the following properties is employed:

- the quantized disparity values are smoothed to remove quantization artifacts; and
- undefined regions in the disparity maps are interpolated.

A multiscale multigrid surface reconstruction algorithm based on the work of Terzopoulos [31] was chosen to interpolate the disparity map with a finite element approximation (see also Volume 2, Chapters 4 and 14). It is assumed that the object area consists of smooth coherent surfaces that can be modeled as a thin plate with a certain stiffness and that inside such a region the disparity measurement is either corrupted by noise or no estimate is available. Care must be taken to select the appropriate noise model for the interpolation of coarsely quantized disparity maps. This can be achieved by incorporating the quantization stepsize in the measurement model [32]. The approach smoothes the quantization steps but preserves object edges.

20.4.5 Subpel disparity estimation

For highest precision a subpixel estimate may be obtained. The interpolated disparity estimates are taken as a starting value for a gradient matching technique with subpixel accuracy. An affine geometric transformation accounts for the perspective warping of the measurement window. It is assumed that the object consists mostly of smooth surfaces with arbitrary surface orientation. A measurement window (of

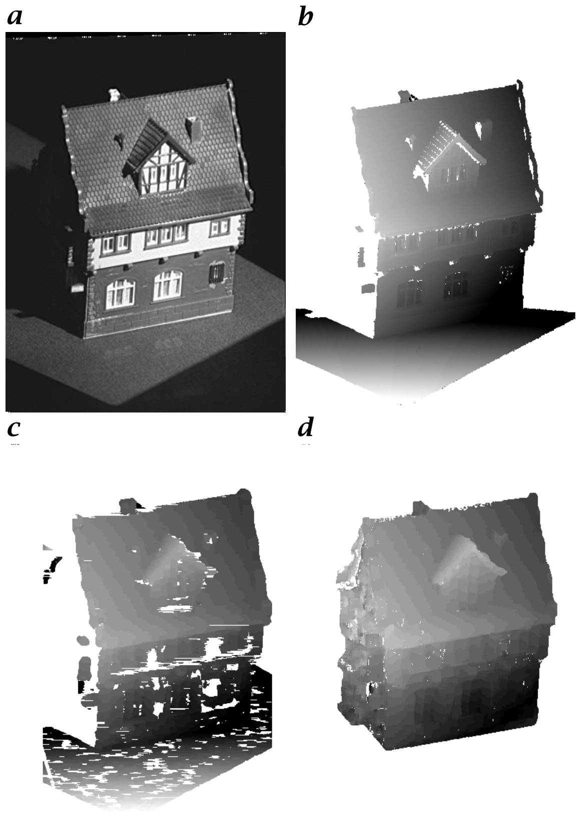


*Figure 20.5: Comparison of depth estimation methods for toy house: **a** left original image; **b** reference depth map (dark = far, light = near); **c** pel-search depth map; **d** subpel-estimation depth map, initialized by the interpolated pel-search depth map.*

11×11 pixel) is used for intensity matching. The difference in gray-level distribution is minimized by fitting the affine transformation between the measurement windows with a robust least squares estimation [33, 34].

20.4.6 Evaluation of measurement accuracy

The performance and accuracy of the discussed approach is important when applied in real measurement situations. To quantify the estimation quality we tested the algorithm in a well-calibrated laboratory setup. A toy house with dimensions $150\,\text{mm}^3$ was measured from a distance of 1300 mm by a calibrated binocular camera rig with baseline distance of 120 mm. The ground truth for this setup was obtained

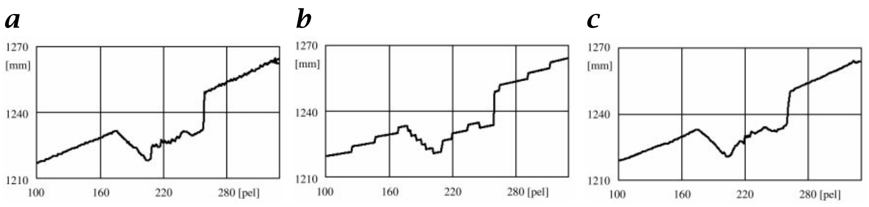


Figure 20.6: *Depth profiles over the roof area with window:* ***a*** *reference profile;* ***b*** *pel-search profile;* ***c*** *subpel profile.*

Table 20.1: *Statistics of the mean error e_{dm}, its standard deviation σe_d and the relative depth error e_{dr} for the house model, estimated over 71,513 valid depth measurements*

Method	e_{dm}[mm]	σe_d[mm]	$e_{dr} = \frac{e_d}{d}$[%]
Reference	0.00	0.39	0.03
1-pel search	0.28	2.3	0.17
Interpolation	0.23	4.1	0.31
Subpel estimate	0.21	1.5	0.11

by a high resolution active light stripe projection unit (Volume 1, Chapter 20, [35]) that produces a reference depth map with a relative depth error of 0.03 %. Figure 20.5 shows one of the input images and the depth maps obtained with the different measurement methods. The results of the disparity search algorithm in Fig. 20.5c was interpolated and taken as initialization for the subpixel estimator.

The depth maps in Fig. 20.5 give a qualitative impression on the obtained accuracy. For a quantitative evaluation we took a depth profile over the object at the height of the roof with its roof window. Figure 20.6 shows the three profiles. The quantization stepsize of the disparity search (2.5 mm/pixel) is clearly visible in profile Fig. 20.5b. In the subpixel estimation the quantization artifacts are removed but some smoothing is introduced at object edges.

Quality measures for the algorithms were obtained by a statistical analysis of the deviation between reference depth and estimated depth. The mean depth error e_{dm}, the standard deviation σe_d and the relative depth error e_{dr} are listed in Table 20.1.

The interpolation introduces some error due to smoothing that is reflected in the higher depth error variance. The subpel estimator performs best. These values, however, do not reflect the visual reconstruction quality. The interpolation removes the quantization artifacts that results in a much better visual appearance of the surface.

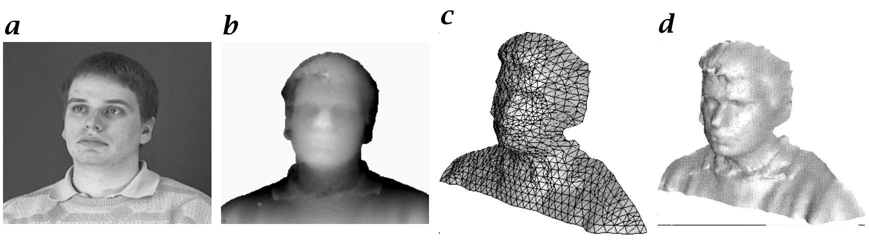


Figure 20.7: *Modeling from image pair:* ***a*** *original image;* ***b*** *interpolated depth map;* ***c*** *3-D wire-frame;* ***d*** *shaded surface model.*

20.5 Three-dimensional model building

The modeling of complex objects requires that more than one view is evaluated to account for occlusions and to improve the object geometry. Surface models from different view points have to be registered into a consistent object coordinate system. The view-dependent depth maps are converted to 3-D surface models and newly visible surface areas are fused into one complete object model. The different modeling steps will be explained with the *Head* sequence where a person rotates on a swivel chair in front of a calibrated stereo rig. A sequence of 160 stereoscopic image pairs was taken during one full rotation of the person.

20.5.1 Three-dimensional surface approximation

The depth maps are converted into a parametric 3-D surface approximation by spanning a triangular wire-frame in space (see also Volume 2, Chapter 25). The triangular mesh was chosen because it is capable of representing arbitrary surface geometries without singularities. The size of the mesh triangles can be chosen such that not too much surface detail is lost while keeping the total amount of triangles low. Figure 20.7 displays modeling results for a single depth map of the *Head* sequence. Only 1 % of the depth map points were kept as mesh vertices. This is sufficient to preserve surface geometry as can be seen in the shaded model. The facial features and even small details such as the collar could be measured.

20.5.2 Intensity-based motion tracking

Feature-based camera tracking and pose registration as part of the camera calibration process were discussed in Section 20.3. If the motion between successive images is small then the pose estimate can be refined by an intensity-based algorithm [36]. The approach will be discussed here for a stationary camera and moving objects but is easily extended to moving cameras [34].

Each object in the scene is treated as an arbitrarily shaped, rigid 3-D surface. Assume that the surface holds the intensity pattern of the real object. A surface point $\boldsymbol{x}$ is projected through a camera projection matrix $\boldsymbol{P}$ onto the image at $\boldsymbol{q}$ with image intensity I_q. As the object point moves from $\boldsymbol{x}$ to $\boldsymbol{x}_T$ with an incremental orientation change $(\boldsymbol{R}_T, \boldsymbol{t}_T)$, the projection causes an optical flow vector $\boldsymbol{f} = \boldsymbol{q}_t - \boldsymbol{q}$ in the image. The object motion and its projected flow vector are governed by the general motion Eq. (20.3)

$$\boldsymbol{f}_q = \boldsymbol{q}_T - \boldsymbol{q} = \boldsymbol{P}(\boldsymbol{x}_T\text{-}\boldsymbol{x}) = \boldsymbol{P}\begin{bmatrix} \boldsymbol{R}_T - \boldsymbol{I} & \boldsymbol{t}_T \\ 0 & 1 \end{bmatrix}\boldsymbol{x} \tag{20.3}$$

$$\text{with} \quad \boldsymbol{x}_T = \begin{bmatrix} \boldsymbol{R}_T & \boldsymbol{t}_T \\ 0 & 1 \end{bmatrix}\boldsymbol{x} \tag{20.4}$$

The optical flow can be measured from temporal image intensity changes ΔI_q and spatial image gradients $\boldsymbol{g}_q$ by the optical flow constraint Eq. (20.5)

$$\Delta I_q = I_{qT} - I_q = \boldsymbol{g}_q{}^T \boldsymbol{f}_q \quad \text{with} \quad \boldsymbol{g}_q{}^T = \left(\frac{\partial I_q}{\partial q_x}, \frac{\partial I_q}{\partial q_y}\right) \tag{20.5}$$

Substitution of Eq. (20.3) in Eq. (20.5) yields a direct relation between the observed image intensity changes and the object motion parameters

$$\Delta I_q = \boldsymbol{g}_q{}^T \boldsymbol{P}\begin{bmatrix} \boldsymbol{R}_T - \boldsymbol{I} & \boldsymbol{t}_T \\ 0 & 1 \end{bmatrix}\boldsymbol{x} \tag{20.6}$$

Equation (20.6) can be computed for each object point that holds image gradients above a noise threshold. The refined pose estimate $(\boldsymbol{R}_T, \boldsymbol{t}_T)$ is computed by using a minimum variance estimator over all measurements. It yields accurate measurements because the estimates are derived directly from the image intensities. Equation (20.6) is iterated to account for nonlinearities and convergence is improved by estimating hierarchical image pyramids [34].

To verify the pose estimator with ground truth information, the toy house (see Fig. 20.5) was placed on a computer-controlled calibrated turntable. Seventy-two images with a table rotation of 5° between each view were recorded with a calibrated stereo rig. The object was modeled from one stereo pair and than tracked over a rotation of about 20°.

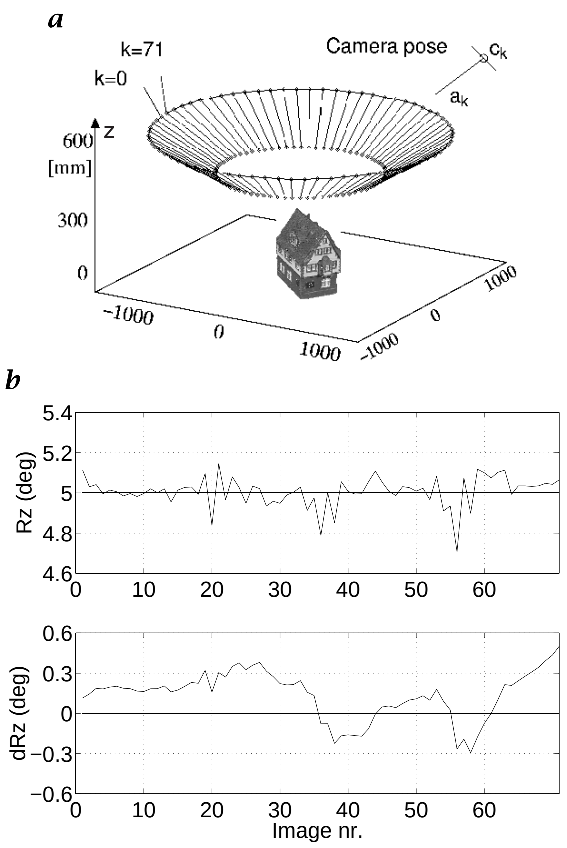


Figure 20.8: ***a*** *Object and estimated camera poses for $k = 72$ images as indicated by projection center $\boldsymbol{c}_k$ and optical axis $\boldsymbol{a}_k$;* ***b*** *estimated rotation angle R_z (true value = 5° and accumulated rotation deviation dR_z over a full rotation (72 views).*

This procedure was repeated every 20° while registering the object pose in turntable coordinates. Figure 20.8 shows the measurement arrangement with the estimated camera poses (a) and the estimated turntable rotation angle R_z. The estimated pose showed very good agreement with the ground truth pose. When tracked over a complete rotation of 360° the maximum angular deviation was below ± 0.5°.

20.5.3 Surface update and depth fusion

While the object rotates in front of the camera, new surface regions appear that need to be modeled, registered and fused to the already modeled surface. Object regions that are visible in more than two views can be updated to increase the surface accuracy.

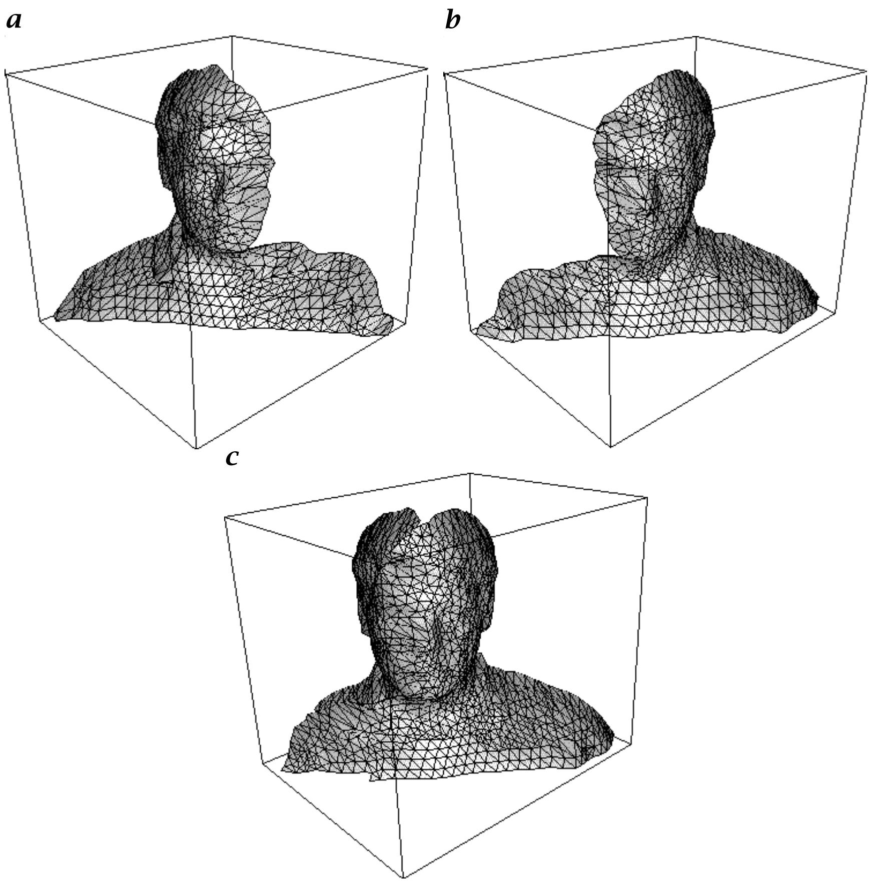


Figure 20.9: *Surface fusion for the head:* ***a*** *left surface;* ***b*** *right surface;* ***c*** *fused surface.*

Surface update. Updating from multiple view points is performed by applying a Kalman filter to each object mesh vertex. The vertex is projected onto the current view point and its 3-D position is updated according to the measured depth and associated measurement uncertainty. As the view point changes the viewing angle increases and measurements become more accurate [37].

Surface fusion. Fusion is needed when object parts overlap partially and new regions become visible. Due to systematic errors and misalignments in the motion tracking the surfaces may not overlap perfectly. Fusion must take care of the misalignment and provide a smooth transition between the surfaces in the overlap area. The gap between both surfaces is closed by weighted depth interpolation. A virtual camera

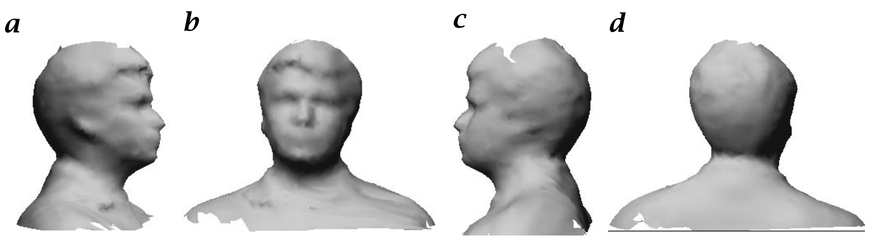


Figure 20.10: *Synthesized views of the shaded head model from* ***a*** *right;* ***b*** *front;* ***c*** *left;* ***d*** *back view.*

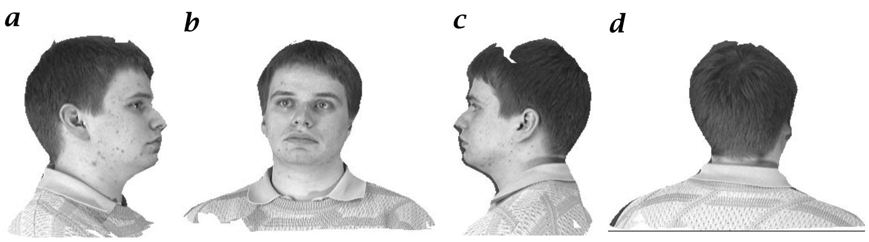


Figure 20.11: *Synthesized views of the textured head model from* ***a*** *right;* ***b*** *front;* ***c*** *left;* ***d*** *back view; for movie see* `/movies/20/head.mov`.

is placed facing the overlap area and the depth of both surfaces to the camera is computed. The fused surface in the overlap area is then interpolated with a weighting function that ensures a continuous and smooth transition between the surfaces.

New object parts are added incrementally patch by patch until the object is completely modeled. Figure 20.9 shows the concatenation of two surfaces from a left and right front view into the fused surface. This fusion is repeated for other view points until the head model is completely closed [38].

Surface texture. Texture is added to the complete model in much the same way as the fusion process. For each surface triangle its visibility in the images is computed. The image with highest texture resolution is selected and a texture map is assigned to each triangle. Inconsistencies in illumination between texture maps of adjacent triangles are smoothed by interpolation. A more complete discussion on this topic can be found in Niem and Broszio [39]. Figures 20.10 and 20.11 display some synthesized views of the complete model from all sides. The model consists of a wire-frame with 3250 vertices and a texture map with 1024×960 pixels. The shaded views show the geometric surface detail while the visual quality can be evaluated in the textured views.

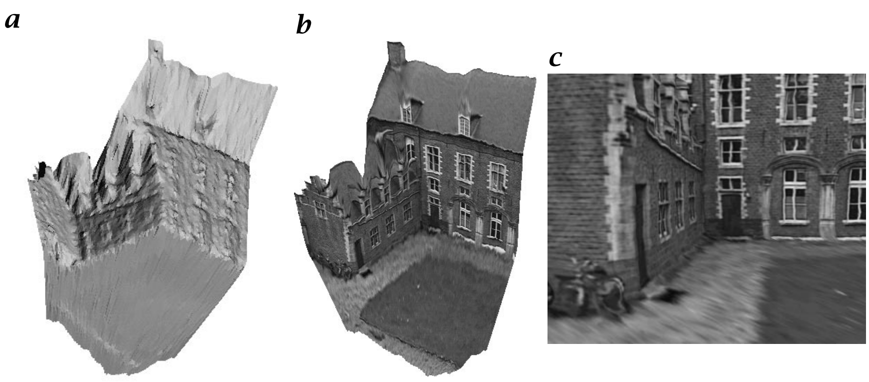


Figure 20.12: ***a*** *Shaded;* ***b*** *textured; and* ***c*** *close-up view of the castle model; for movie see* /movies/20/castle.mov

20.6 Uncalibrated monocular sequences

In this section, the performance of the modeling system is tested on different monocular outdoor sequences. The *castle sequence* was introduced in Section 20.3.2 to demonstrate point tracking and camera calibration. To judge the geometric and visual quality, a model is reconstructed from the image sequence in Fig. 20.3 and different perspective views of the model are computed and displayed in Fig. 20.12. In the shaded view, the geometric details such as the window and door niches are seen. A close-up look from a position that a human observer would take reveals the high visual quality of the model.

A more quantitative evaluation of metric properties was obtained by measuring angles in the reconstructed scene between parallel lines ($1.0 \pm 0.6^\circ$) and orthogonal lines ($92.5 \pm 0.4^\circ$). These results confirm the good metric reconstruction obtained by the method (see [21]).

20.6.1 Three-dimensional modeling at Sagalassos: A test case

The system was tested on a variety of scenes with different cameras of varying quality (35 mm photo camera on Photo-CD, digital still camera, camcorders) and was found to work even in difficult acquisition circumstances. As special test case field trials were carried out at the archaeological excavation site of Sagalassos in Turkey. This is a challenging task because the archaeologists want to reconstruct even small surface details and irregular structures. Measurements with highly calibrated photogrammetric workstations failed because those systems could not withstand the high temperatures at the site. We show here two of the different scenes selected for modeling.

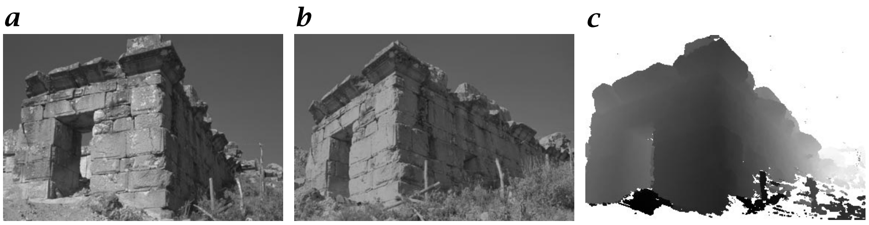


Figure 20.13: ***a*** *Image 3 and* ***b*** *image 6 of the Roman bath sequence;* ***c*** *estimated depth map for image 6 (near = dark, far = light).*

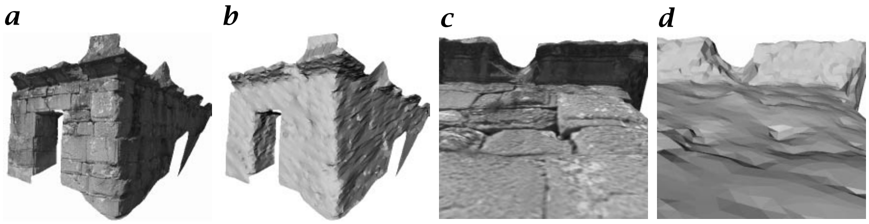


Figure 20.14: ***a*** *Textured and* ***b*** *shaded view of the Roman bath model. The close-up view in* ***c*** *(textured) and* ***d*** *(shaded) shows that even small details such as single stones are modeled; for movie see* `/movies/20/bath.mov`.

Roman bath. This scene shows the reconstruction of parts of the *Roman bath* at Sagalassos from eight uncalibrated images taken with a standard photo camera. Figure 20.13 shows two of the images and the corresponding depth map. The relative depth error was estimated to 0.8 % and the depth map is very dense. Figure 20.14 reveals the high reconstruction quality. The model gives a realistic impression of the real object. The close-up view reveals that each stone is modeled, including reliefs and small indentations. The indentations belong to the fittings between the stones.

Sagalassos site. The *Site* sequence in Fig. 20.15 is a good example of a large-scale modeling using off-the-shelf equipment. Nine images of the complete excavation site of Sagalassos in Turkey (extension of a few square kilometers) were taken with a conventional photographic camera while walking along the hillside. The film was then digitized on PhotoCD. The *Site* model in Fig. 20.16 gives a good reconstruction of the valley relief. Some of the dominant objects such as the Roman bath and the Agora, as well as landmarks such as large trees or small hills are already modeled at this coarse scale. It is intended to register the detailed object models like the Roman bath to the Site and to build a virtual excavation site that one can visit as part of a virtual archaeological showcase.

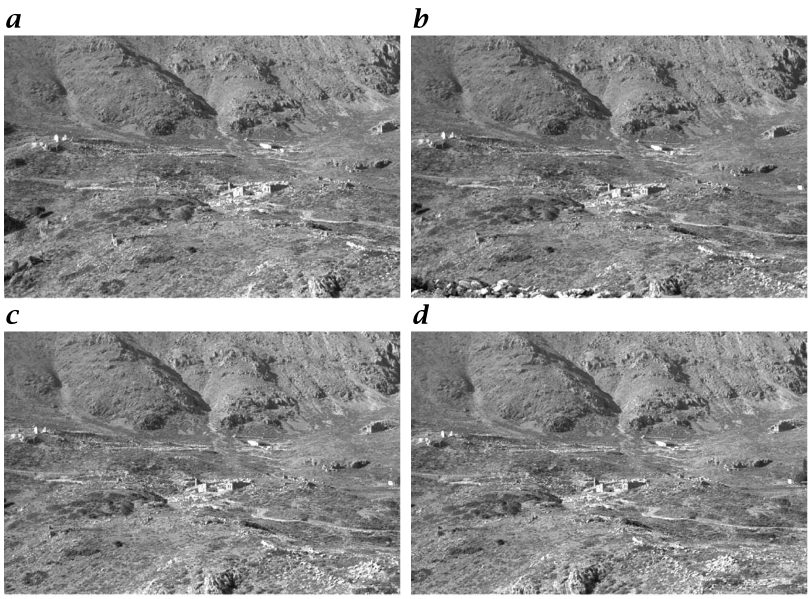


Figure 20.15: *Four of the nine images of the Site sequence.*

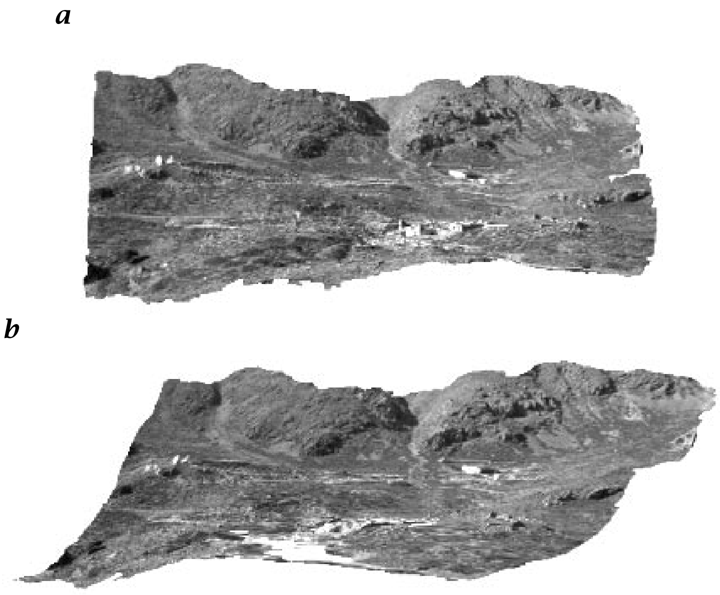


Figure 20.16: *Textured views of the site reconstruction; for movie see* `/movies/20/baths.mov`*; additional movies:* `/movies/20/site.mov` *and* `/movies/20/temple.mov`*.*

20.7 Conclusions

An automatic 3-D scene modeling system was presented that is capable of building complete surface models from image sequences. The system handles calibrated stereoscopic sequences as well as uncalibrated monocular sequences. It computes dense depth maps and textured 3-D surface models and the object and camera motion from the image sequence. The system successfully models scenes assuming piecewise-smooth surfaces. However, some work remains to be done to engineer a really robust modeling system for everyday use.

Acknowledgments

The system has gained much strength and flexibility by the extension towards uncalibrated sequences. This was made possible by the work of Marc Pollefeys and Luc Van Gool (K.U.Leuven/Belgium) on self-calibration and by the contributions from Andrew Zisserman and his group at Oxford on robust point tracking and projective structure from motion. The financial support of research grants from the German TELEKOM, the EU-ACTS Project VANGUARD and the Belgian Project IUAP IV-24 are gratefully acknowledged.

20.8 References

[1] Durisch, P., (1992). Photogrammetry and computer graphics for visual impact Analysis in Architecture. In *Proc. ISPRS Conference*, L. W. Fritz and J. R. Lucas, eds., Vol. 29,B5, pp. 434–445. Bethesda, MD: ASPRS.

[2] Blonde, L., (1994). The MONA LISA project: general presentation. In *Proceedings on the European Workshop in Combined Real and Synthetic Image Processing for Broadcast and Video Production*, Y. Paker and S. Wilbur, eds., Workshops in Computing, pp. 3–12. London: Springer.

[3] Harashima, H. and Kishino, F., (1991). Intelligent image coding and communications with realistic sensations—recent trends. *IEICE Transactions*, **E 74 (6)**:1582–1592.

[4] Aloimonos, J. and Shulman, D., (1989). *Integration of Visual Modules*. San Diego, U.S.A: Academic Press.

[5] Jarvis, R. A., (1983). A perspective on range finding techniques for computer vision. *IEEE Trans. PAMI*, **5 (2)**:122–139.

[6] Aggarwal, J. and Nandhakumar, N., (1988). On the computation of motion from sequences of Images—A Review. In *Proc. IEEE*, Vol. 76, pp. 917–935. Los Alamitos, CA: IEEE Computer Soc. Press.

[7] Netravali, A. and Salz, J., (1985). Algorithms for estimation of three-dimensional motion. *AT&T Technical Journal*, **64(2)**:335–346.

[8] Terzopoulos, D. and Metaxas, D., (1991). Dynamic 3D models with local and global deformations: deformable superquadratics. *IEEE Trans. PAMI*, **13**:703–714.

[9] Pentland, A. and Horowitz, B., (1991). Recovery of nonrigid motion and structure. *IEEE Trans. PAMI*, **13**:730–742.

[10] Tomasi, C. and Kanade, T., (1992). Shape and motion from image streams under orthography: A factorization method. *Int. Journal on Computer Vision*, **9**:137–154.

[11] Matthies, L., Kanade, T., , and Szeliski, R., (1989). Kalman filter-based algorithms for estimating depth from image sequences. *International Jour. Computer Vision*, **3**:209–236.

[12] Robert, P. and Ogor, F., (1994). Joint estimation of depth maps and camera motion in the construction of 3D models from a mobile camera. In *Proceedings on the European Workshop in Combined Real and Synthetic Image Processing for Broadcast and Video Production*, Y. Paker and S. Wilbur, eds., Workshops in Computing, pp. 147–163. London: Springer.

[13] Harris, C. and Pike, J., (1987). 3D positional integration from image sequences. In *3rd Alvey Vision Conference*, pp. 233–236.

[14] Negahdaripour, S., Hayashi, B., and Aloimonos, Y., (1995). Direct motion stereo for passive navigation. *IEEE Trans. Robotics and Automation*, **11(6)**: 829–843.

[15] Tirumalai, A., Schunck, B., and Jain, R., (1992). Dynamic Stereo with Self-Calibration. *IEEE Trans. PAMI*, **14**:1184–1189.

[16] Zhang, Z. and Faugeras, O., (1992). *3D Dynamic Scene Analysis*. Heidelberg, New York: Springer.

[17] Faugeras, O., (1992). What can be seen in three dimensions with an uncalibrated stereo rig? In *Proc. ECCV 92*, G. Sandini, ed., Vol. 588 of *Lecture Notes on Computer Science*, pp. 563–578. Berlin: Springer.

[18] Hartley, R., (1992). Estimation of relative camera positions for uncalibrated cameras. In *Proc. ECCV 92*, G. Sandini, ed., Vol. 588 of *Lecture Notes on Computer Science*, pp. 579–587. Berlin: Springer.

[19] Heyden, A. and Åström, K., (1997). Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In *Proc. CVPR97*, pp. 438–443. Los Alamitos, CA: IEEE Comp. Society.

[20] Pollefeys, M. and VanGool, L., (1997). Self-calibration from the absolute conic on the plane at infinity. In *Proceedings of Int. Conf. CAIP, Kiel*, G. Sommer, ed., Vol. 1296 of *Lecture Notes on Computer Science*, pp. 175–182. Berlin: Springer.

[21] Pollefeys, M., Koch, R., and VanGool, L., (1998). Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters. In *Proc. 6th ICCV'98, Bombay*, pp. 90–95. New York: IEEE.

[22] Triggs, B., (1997). The absolute quadric. In *Proceedings of CVPR97*, pp. 609–614. Los Alamitos, CA: IEEE Comp. Society.

[23] Koch, R., (1995). 3-D surface reconstruction from stereoscopic image sequences. In *Proc. 5th International Conference on Computer Vision, Boston, U.S.A.* Los Alamitos, CA: IEEE Computer Society Press.

[24] Tsai, R., (1985). A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. In *RC 11413*. Yorktown Heights, NY: IBM Research Report.

[25] Beardsley, P., Torr, P., and Zisserman, A., (1996). 3D model acquisition from extended image sequences. In *Computer Vision—ECCV 96*, B. Buxton and R. Cipolla, eds., Vol. 1064 of *Lecture Notes in Computer Science*, pp. 683–695. Berlin: Springer.

[26] Torr, P., (1995). *Motion Segmentation and Outlier Detection.* PhD thesis, Dept. Eng. Science, University of Oxford, U.K.

[27] Kochan, A., (1991). *Eine Methodenbank zur Evaluierung von Stereo-Vision-Verfahren.* Dissertation, TU Berlin.

[28] Cox, I., Hingorani, S., Maggs, B., and Rao, S., (1992). Stereo without regularisation. In *British Machine Vision Conference*, D. Hogg and R. Boyle, eds., pp. 337–346. London: Springer.

[29] Falkenhagen, L., (1997). Hierarchical block-based disparity estimation considering neighbourhood constraints. In *Proc. International Workshop on SNHC and 3D Imaging*, N. Serris and M. G. Strintzis, eds., pp. 115–118. Brussels, Belgium: European Commission, ACTS057-Vidas.

[30] Gimel'farb, G. L., (1979). Symmetrical approach to the problem of automating stereoscopic measurements in photogrammetry. *Cybernetics, Consultants Bureau, N.Y*, **15(2)**:235–247.

[31] Terzopoulos, D., (1988). The computation of visible-surface representations. *IEEE Trans. PAMI*, **10**:417–438.

[32] Koch, R., (1996). Surface segmentation and modeling of 3-D polygonal objects from stereoscopic image pairs. In *Proc. ICPR96, Vienna.* Los Alamitos, CA: IEEE Computer Society Press.

[33] Ackermann, F., (1984). Digital image correlation: Performance and potential applications in Photogrammetry. *Photogramm. Record*, **11(64)**: 429–439.

[34] Koch, R., (1997). *Automatische Oberflächenmodellierung starrer dreidimensionaler Objekte aus stereoskopischen Bildfolgen*, Vol. 499 of *Fortschr.-Berichte Reihe 10.* Düsseldorf: VDI Verlag.

[35] Wolf, H., (1992). *Schnelle 3-dimensionale Bilderfassung mit dem codierten Lichtansatz*, Vol. 939 of *VDI-Berichte.* Düsseldorf: VDI Verlag.

[36] Koch, R., (1993). Dynamic 3D scene analysis through synthesis feedback control. *IEEE Trans. PAMI*, **15**:556–568.

[37] Koch, R., Pollefeys, M., and VanGool, L., (1998). Multi viewpoint stereo from uncalibrated video sequences. In *Computer Vision - ECCV'98*, H. Burkhardt and B. Neumann, eds., Vol. 1406/1 of *Lecture Notes in Computer Science*, pp. 55–71. Heidelberg: Springer.

[38] Koch, R., (1996). 3D modeling of human heads from stereoscopic image sequences. In *Proc. DAGM'96*, B. Jähne, P. Geißler, H. Haußecker, and F. Hering, eds., Informatik Aktuell, pp. 169–178. Berlin: Springer.

[39] Niem, W. and Broszio, H., (1995). Mapping texture from multiple cameras onto 3-D Object Models for Computer Animation. In *Proc. Int. Workshop on Stereoscopic and Threedimensional Imaging, Santorini, Greece*, S. Efstratiadis and M. G. Strintzis, eds. Brussels, Belgium: European Commission, RACE-2054-DISTIMA.

21 Viro-3D—Fast Three-Dimensional Full-Body Scanning for Humans and Other Objects

Norbert Stein, and Bernhard Minge

VITRONIC Dr.-Ing. Stein Bildverarbeitungssysteme, Wiesbaden, Germany

21.1 Introduction

To perform fast and complete 3-D scans on human beings or other living objects is a challenge very often facing 3-D measuring technology. The development of multisensor systems operating on the basis of the laser light-stripe method has proved to be an effective means to fulfill this task. Within seconds human beings and other objects can be measured three-dimensionally.

A number of different methods have already been developed to generate 3-D measured data. In one of the most common methods mechanical 3-D coordinate measuring machines are used. A probe is advanced

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

towards the object by means of a precision mechanical system. At the moment the probe contacts the object to be scanned, the coordinate position of the machine arms are interrogated and the coordinates of the contact point are calculated. Another method is based on the same principle, but uses an optical probe to perform contact-free measurements. Due to the fact that both systems require several seconds to record each 3-D measured value, they are only suited to scan static objects. This also applies to the method of photogrammetry in which a series of photographs are taken from different angles. The depth information is obtained by means of control points or clearly identifiable corners and edges.

The measuring principle for fast 3-D information recording basically corresponds to the triangulation principle as described in Chapter 18 and is, thus, a method for measuring distances or heights. The camera and the illumination system are positioned on the same side of the object to be scanned. The elements are arranged relative to each other, so that the scene to be assessed is illuminated with a laser while a camera is aimed at the laser plane at a permanent defined angle to the laser beam. General information about illumination systems in Chapters 3 and 6.

Other optical methods that are based on the generation of special light patterns (for more information about light-stripe projection, phase-shift method and structured illumination, see also Chapter 20) depend considerably on the surface condition (reflection) of the object. In order to perform a measurement several images have to be taken consecutively. During the measurement process the object has to be static.

Some of the forementioned methods are thus unsuitable for measuring nonstatic objects such as human beings. In contrast to them the split-beam or light-stripe method is particularly suitable for 3-D profile scanning on living objects and thus forms the basis of the measuring principle used in the full-body scanner.

21.1.1 Light-stripe scanners

The light source usually employed within this process consists of a laser light source in combination with a cylindrical lens. A video camera is positioned at a defined angle to the light source (Fig. 21.1). The recorded image contains the object information for a complete profile section over the entire width of the image field. The lateral assignment of measured values is thus unique, as this information is present without an additional shift in the measured image.

The light line, imaged on the object, is located in the field of view of the camera and is offset and deflected in accordance with the different heights of the object. The angle between the camera and the light

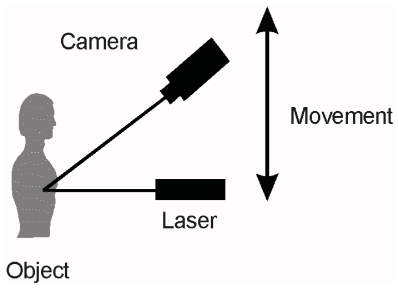


***Figure 21.1:** Sketch of the principle of full-body scanning:*

source (triangulation angle) determines the magnitude of the offset and the deflection. The greater the angle, the greater the offset. As the angle is given, the shape of the object at the respective line position can be determined by means of a simple measurement, and a profile section can be generated. Each sensor thus consists of a line-light source (diode laser with a cylindrical lens) and a matrix video camera with an upstream bandpass filter matched to the laser. By using the laser-filter combination, in many applications the sensor systems are predominantly insensitive to external influences. For general information about lenses and optics see Chapter 4.

If the sensor is moved over the object and profile sections are generated simultaneously, a 3-D image of the outer contour of the object is produced. The measured data obtained in this way encode the spatial coordinates x, y, z of the laser's line of incidence on the object. The smallest distance possible between two consecutive profile sections at a given sensor speed is achieved when the height profiles are recorded in video real-time. For charge coupled device (CCD) cameras that fulfill the requirements of the European TV standard, that means that the images are recorded every 20 ms. In this way, 50 profile sections/s can be generated. With special cameras also 1000 Hz can be achieved.

It goes without saying that one sensor is insufficient to scan an entire body as it can only "see" those parts of the object that are in its field of view. For this reason, several sensors have to be combined in order to sight the object from different directions. Depending on the desired measuring resolution and the size of the object to be scanned, it may also be necessary to divide the area into several camera fields. The individual cameras have to be matched by means of a special calibration process so that the measured data can be combined.

21.1.2 General scanner requirements

Regardless of the method used, the optical path must not be disturbed when the signal is recorded. This applies to both shading effects caused by the measured object itself (undercuts), as well as for the surrounding parts that cause interference in the image field.

It should be noted that the inclination of the object surface to be tested has an effect on the quality of the signal (reflections). Depending on the selected resolution and the precision of the measuring system, the roughness of the surface and waviness can also have an effect.

With transparent surfaces, it must also be determined whether signals can be recorded at all. The effects of light from external sources can be counteracted by increasing the strength of the light source to such an extent that a suitable useful signal can still be received using optical filters. For this reason, it can be concluded that systems using laser light are preferable as long as they are eye-safe. If the intensity of the interfering light in the spectral range of the useful signal is too high, a solution may be provided by changing the frequency of the light used. Direct external light, such as broadband interfering radiation (e.g., sunlight), should be avoided if possible, so that the required light output of the illumination system is not unnecessarily high (safety aspects for lasers).

With every application, it should be noted that the resolution or the obtainable precision and measurement volume are directly related. The larger the measurement volume examined with a sensor, the lower the maximum obtainable resolution of the measuring system. The resolution can only be increased above the actual pixel resolution (subpixel range) if the relevant ambient and measurement requirements are fulfilled.

21.2 Evaluation hardware and software

To generate the whole measuring volume at a time, all sensors have to be connected in parallel to the evaluation unit (multisensor system). For this purpose, high-performance processors are combined with real-time image processing chips that were specially designed for VIRO-3D systems. These image processing chips allow the recording of the image data from practically any number of sensors and to evaluate it synchronously in video real-time. They supply 3-D raw data for each sensor, which is stored in the main memory of the computer and processed by the processor. The 3-D raw data are then converted from pixel to a metric unit and undergoes a mathematical regression analysis of object-dependent order, so that a smooth signal form can be generated. The measured data from all section planes are combined to form a 3-D model. Depending on the type of application, an additional smoothing, motion compensation, conversion of the point scatter to computer-aided design (CAD) files, or a pseudo 3-D representation can be carried out.

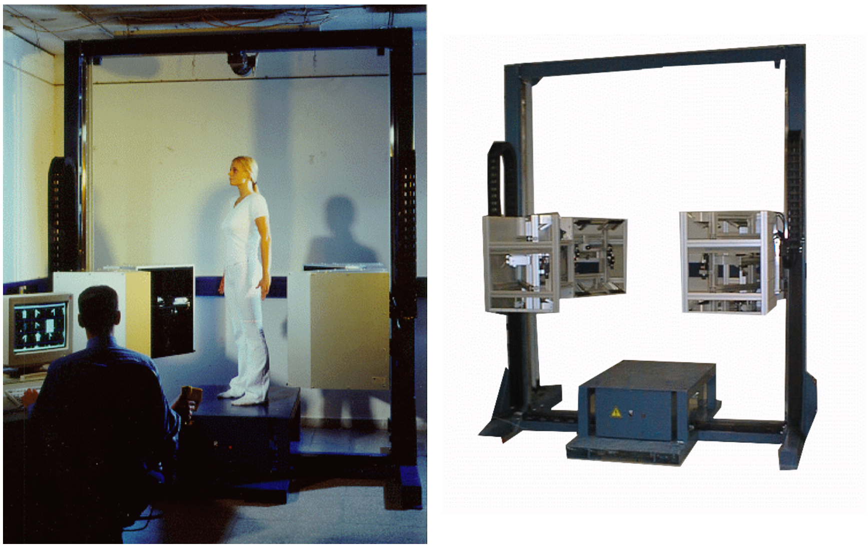


Figure 21.2: *Full-body scanner VITUS.*

21.3 Mechanical design

For a complete 360° scan of an object an all-round sensor configuration is necessary. Generally, three different travel directions are possible: linear, which can be both vertical and horizontal, or rotary. The latter are the most economical solution, but can result in measuring errors due to unavoidable, spontaneous movements of the body. Because this leads to incorrect measurements in each measuring plane, the final result can be useless. For this reason vertical feed was chosen, so that any spontaneous movement of the body has only minimum effect on the measurement and can be detected and edited out of the measured data.

In order to scan a high measuring volume, sufficiently long and stable mechanical systems are required. The setup of the VITUS 3-D scanner is shown in Fig. 21.2. The line lasers are mounted on a fully adjustable sensor platform on the sensor frame and are aligned in a way that the laser lines are horizontal and the section planes are flat. The cameras are secured to adjustable mounts in the same way.

In order to guarantee a high line quality required for a high measuring resolution, the area to be measured can be divided into different fields. In addition to that, the measuring area is divided into several sections by different cameras. They are located above the lasers and cover a precisely defined triangulation angle. In case of undercuttings, additional cameras are positioned underneath the laser and cover a comparable measuring range.

21.4 Measuring process

Once the measurement starts, the light sources are switched on. During the measurement, the slide travels vertically at nominal speed and is nearly vibration-free. A rotational angle sensor supplies clock signals to the evaluation unit, so that the sensor position can be calculated for each individual section measurement.

The cameras continuously provide image signals, which are evaluated in real-time. The traveling speed of the slide can be set by the user for each individual measurement. It directly influences the measuring time, the section density and, thus, the system's resolution in vertical plane. Once the measuring time has expired, the scanned person can leave the gantry and within a few seconds the system is ready to perform the next measurement.

An example helps illustrate the complexity of this measuring process. By means of a VIRO-3D scanning system it takes 18 s to scan a person with a size of 1.8 m. During this period of time 900 360° profile sections are produced. Each section measurement consists of nearly 7000 individual measurement steps, so that within 18 s more than 6 million 3-D measuring points are processed, which then form a 3-D model made of 0.7-1 million 2-D contour points. These 6 million 3-D points are generated from 4.5 billion image points (4.5 GByte).

21.5 Ranges of application

Most of the scanning systems basically follow the same design and measuring principles, but offer a range of different measuring volumes and resolutions by using different numbers of lasers and cameras. Even different focal lengths can be used. After the system has been set up, the sensors have to be adjusted and calibrated precisely (Chapter 19). For this purpose, a special calibration process has been developed that is carried out automatically. In the described applications, the following resolutions depending on the setup are achieved.

Actually, the system will be used for studies in the clothing industries and ergonomical products. All these applications need the 3-D information of parts or complete human bodies for creating made-to-measure products on an actual data basis. Tests with made-to-measure clothing straight from the factory are still going on. The target was automatic 3-D scanning of the customer's contours in combination with automatic generation of patterns that could be used to produce made-to-measure clothes and shoes. In some of these applications a link with the men-model RAMSIS was created to use also the RAMSIS features (connection to CAD programs, simulations, feature extraction, etc.).

Table 21.1: *Examples for ranges of resolution in applications*

Resolution range	Application
>= 1 mm	Measurement of complete human beings
	Measurement of wooden surface profiles
0.1 mm - 1 mm	Measurement of parts of human beings
	Measurement of aluminium ingots
	Welding seam inspection of steel wheels
	Piston assembly
0.01 mm - 0.1 mm	Welding seam Inspection of laser-welded products
	Inspection of tubes for the tightening of seat belts
	Visual inspection of applied solder paste

Systems performing full 3-D scans will be used to create blanks within seconds. After the measured data have been examined and, if necessary, preprocessed, it is converted into a format that is compatible with computerized numerical control (CNC) milling machines. A model is then produced on the milling machine, which in turn can serve as a basis for further blanks. For example, for an artist or industrial sculpture producers support the case for needed milled prototypes.

These systems also can open up new perspectives in various different fields. They could be used in the field of orthopedic medicine to scan and monitor spine curvatures quickly and extremely precisely. Data for made-to-measure orthopedic shoes could be determined with the same degree of ease and accuracy as for special bandages required by fire victims, who will not suffer any discomfort as the measurements are contact-free. Plastic surgery is a further point on the list of future applications.

21.5.1 Scanning of human beings

New impulses will be given to art and photography—sculptures and busts could be created automatically and new methods of portrait studies could be made possible with 3-D photographs (Fig. 21.3). In movies, actors are already emulated and animated in virtual realities. Databases for actors could make the borders between reality and animation more fluid.

21.5.2 Measurement of wooden surface profiles

During the production of chipboards, wooden particles that are mixed with dry glue are strewed on conveyors passing by underneath the

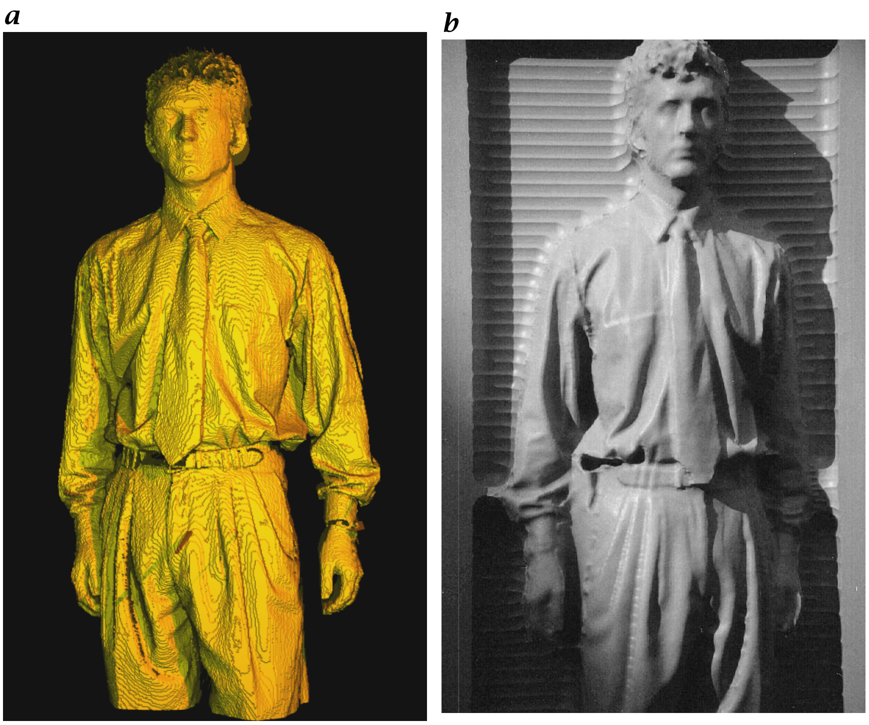


Figure 21.3: ***a*** *Scanning result;* ***b*** *CNC milling result.*

strewing bunkers. The chipfleece is transported into presses and is then pressed into plates. A constant diameter of the chipboards and an even surface must be granted for a high quality of the boards. A light-stripe system is used to measure the elevation of the chipfleece after pressing, so that a 3-D image of the profile of the chipboards is produced. The produced data are immediately transmitted to the data processing unit for a direct regulation of the strewing machines.

A resolution of the elevation-measuring system of 1 mm is required at a strewing width (width of the path) between 1.10 m and 3 m. The cycle rate of the measurement has to be better than 2 profile images per s. The surrounding conditions in the chipboard production are determined by the huge amount of dust, especially at the strewing bunker where the system is installed.

The system was installed as a multicamera system with cameras positioned vertically above the conveyor. The system searches for the lighting model on the fleece, which is heavily disturbed due to the inhomogeneous surface (particles), and decides which profile section of the chip-fleece according to the situation of the illumination model is taken. The machine is installed close to the strewing machines and

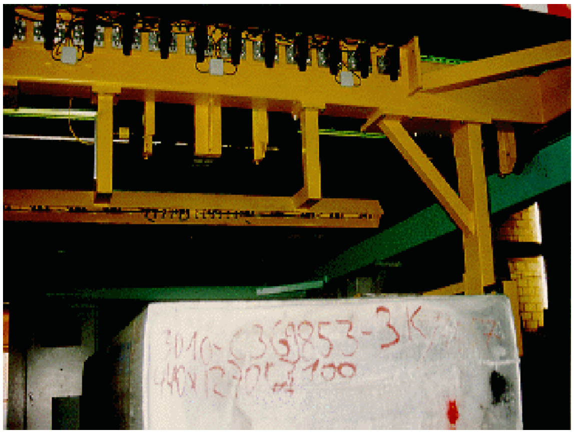


Figure 21.4: *Measurement setup of aluminum ingots.*

even heavy dust development does not cause any disturbances within the production. The customer integrates the system as a standard measuring installation in Europe and overseas.

21.5.3 Measurement of aluminum ingots

An aluminum foundry supplies ingots that are rolled to aluminum sheets or foils at the customer's rolling mill. The 3-D shape of the ingots (dimensions: length: 2.5 - 6.5 m, width: 1 - 2.2 m, height: 0.4 - 0.8 m) is measured by a light-stripe system before the treatment. This is necessary to determine the measurement values, indispensable for the adjustment of the milling machine that mills the ingots, and to get relevant quality parameters of the ingots. For the entire measurement of the ingot a maximum time of 15 s is available. The measurement tolerance for the determination of the ingots maximum and minimum is 0.25 mm, for the measurement of the average of the quality parameters 0.1 mm.

The system operates according to the described principle, whereby light stripes (upper side, 2 lateral faces) are provided by 20 diode lasers with a line-producing optic. The image acquisition is then received by 20 charge coupled device (CCD) matrix cameras (Fig. 21.4). The camera/laser combination relies on a measuring gantry whose cross member is moving across the ingot at a speed of approximately 600 mm/s, during the measurement procedure. The measurement is carried out during the entire procedure including the acceleration and the braking procedure. Here, 20 cameras are analyzed in parallel in video real time. Consequently, the image acquisition consists of 750 3-D profiles with 6000 measuring points per line (approximately 4.5 mill points). After

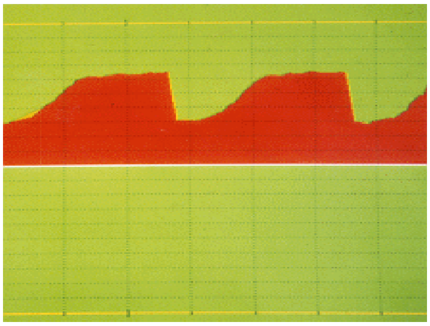


Figure 21.5: *Measured sawtooth profile.*

the measurement, the ingot is milled on the upper side and the lateral faces. Then the ingot is turned over and enters the measuring gantry upside down. Now, the front side of the ingot is also measured.

In order to handle the enormous quantity of data of 15,000 images in the given time, corresponding to 3 gigapixels per measurement, the described special hardware is being used. For each real-time board, five cameras are analyzed in parallel in video real-time. A normal CPU board calculates the quality parameters and controls the milling information as well as the quality report printed as diagrams.

The system is integrated into the production process of a rolling mill in Australia directly in front of the mill. Therefore, outside illumination, extreme atmospheric humidity and temperature, vibrations and aluminum cuttings have to be considered.

21.5.4 Inspection of tubes for the tightening of seat belts

The inner wall of tubes ensuring the tightening of seat belts has a sawtoothed profile (Fig. 21.5). In case of an accident, a bolt that tightens the belt is thrown into the tube. It wedges in the sawtoothed profile so that the belt remains tighten.

This sawtoothed profile is necessary for the function of the belt tightening. The profile had been rolled into the tube sheet metal before the sheet is formed into a tube and then welded. The light-stripe system has to ensure a constant quality of the profile not only during the production of the tube but also during further procedures. Immediately after the rolling stage the system controls the depth of the profile and, at the same time, a direction information that is located on the outside of the tube that has to avoid an incorrect installation of the tube.

The maximum movement speed of the sheet during the profiling is 60 m/min. The sawtoothed profile is embossed into the sheet with a nominal depth of 0.35 mm. A depth of the profile of <0.25 mm is insufficient for the embossing and the system stops the production process. The measurement tolerance in this application is 15 μm.

The installation is directly integrated into the rolling pass of a metal-treating company, where the system is placed partly over and partly under the sheets to be rolled. Shocks, vibrations, dirt, etc., are typical for these surroundings. Six cameras positioned above the measuring point are acquiring the image of the light-profile striking on the sheet. The system controls in video real-time whether or not the sawtoothed profile-line satisfies the requirements. As soon as the required minimum depth is not given the system sends a signal that stops the production.

21.5.5 In-line welding seam inspection of steel wheels

During the automated production process of passenger car wheels the welding seams that are connecting the wheel's bowl and rim have to be inspected automatically according to the following criteria:

- missing welding seam;
- welding seam's length;
- porous welding seam;
- root penetration and undercut;
- seam's position relative to the wheel's bowl;
- thickening at the welding seam's start position;
- unfilled end crater; and
- 3-D measurement of the seam's height.

Extremely different welding seams reflections have to be taken into consideration as soot and smoke areas can alternate with glossy areas on one and the same seam. In this application up to four sensors operating in parallel are used per station (Fig. 21.6). The feeding rate while welding the wheels and while checking the welding seams amounts to 1.2 m/min. This means that profile scans are generated in a raster of 0.4 mm per welding seam. The height resolution as well as the resolution orthogonal to the moving direction is 0.2 mm.

The individual types of wheels are presented for measurement. Prior to the actual inspection, the corresponding inspection program has to be selected. Due to the enabling signal for the inspection system, the wheels are rotated under the sensors over the entire length of the welding seam. With a cycle of 50 Hz the system generates a light stripe at every single welding seam. These successive height profiles of the individual welding seams are combined to a "3-D shape reconstruction" and, according to a corresponding error list, evaluated via different methods of analysis. At the same time, an in-line signal check of the

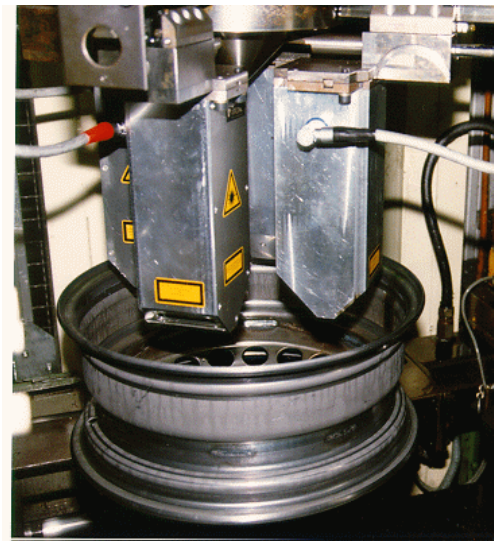


***Figure 21.6:** Measurement setup of welding seam inspection of steel wheels.*

profile information is started. An optimum signal quality considering the seam's reflection features is guaranteed at any time.

To determine the length of the welding seam an external path-indicating signal that detects the beginning and the end of the welding seam is linked via the inspection system. At the end of a working cycle all required details on the welding seam are available and a good/bad signal is transmitted to the main control unit. Additionally, statistics about the type of error as well as the error rate are listed. The operator can call this separate menu on request in order to optimize the welding process.

Upon validation of each individual inspection system in the production environment and an extensive testing phase to prove a 100 % error recognition in comparison with manual inspection by personnel (considering the pseudo-rejection rate, of course), the systems are released as reliable inspection devices and are operating completely self-sufficiently.

Incorrect decisions caused by different assessments, fatigue or daily changes of the personnel occur inevitably when conducting manual inspections, but can be eliminated from now on. This investment helps to increase the quality of the products.

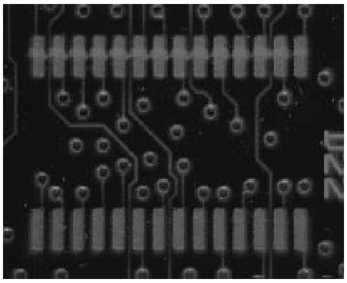


Figure 21.7: *Light stripe on printed circuit boards at the upper row of solder pads.*

21.5.6 Visual inspection of applied solder paste on printed circuit boards

The high density of surface-mounted devices (SMD) on printed circuit boards together with an increasing amount of connection pins per chip leads to an increasing density of pads on SMD printed circuit boards. To solder the SMD components, a solder paste is deposited in a screen printing process before assembling the components on the printed circuit board. Proper, complete, and accurate positioning of the solder paste deposit on the pads is a prerequisite for defect-free assembling of the printed circuit boards. Otherwise, short-circuits or open connections can ensue.

The printed circuit boards are fed by a conveyor belt to the test and inspection station. Once the printed boards are located in the test and inspection station, the inspection system moves to the areas to be inspected. The image field of the camera has been chosen in such a way that entire rows of pads can be inspected within a single image field.

In association with the application-specific flash light system, which can provide different levels of lighting, dependent on the test to be carried out, the system test and inspection station checks to determine if the pads are fully coated (printed) with solder paste and establishes if there is any misalignment in the printed image. Furthermore, the integrated light-stripe system module measures the height of the solder paste deposit at specific places on the board (Fig. 21.7).

Inspecting the entire row of pads is accomplished in < 1 s. After approximately 100 ms, the camera moves to the next position, so that a 100 % inspection of all high-density contacts is performed within the production cycle. The smallest zones of defects or any electrical bridging of just a few hundreds of a millimeter are detected. Testing the areas printed and determining any misalignment of the image are carried out with a repetitive accuracy of just a few microns.

The system detects defective printing with a high degree of certainty. At the same time, an analysis of trends is possible in order to detect in good time any developing defects that may become more serious in the course of time and to eliminate the defects before they adversely affect the quality of the printed circuit board. The defects are highlighted on the monitor thus enabling the operator to visually evaluate the defects.

21.5.7 Piston assembly

Two light-stripe inspection systems ensure the correct assembly of piston rings and guard rings in an assembly line. Both systems use the triangulation measurements to identify and locate the relevant parts with high accuracy. In partly automated piston assembly lines, there are two critical manufacturing steps:

- the correct assembly of all three piston rings of each piston; and
- the correct assembly of the piston and its rod by properly securing the gudgeon pin with guard rings.

Both steps are done manually using dedicated mounting tools. Because these tools can neither ensure that correct ring types are mounted nor detect mispositionings, both steps are highly error prone. The problem is further aggravated due to the fact that deviations between incompatible piston rings may be extremely small, making it impossible for the worker to check the assembly reliably by visual inspection. In addition, the production cycle of five pallets per min, which means 20 pistons per min, respectively, is far too fast for manual inspection by the worker.

To cope with these problems, two inspection stations have been integrated into the manufacturing line, both equipped with the described optical inspection systems. The first station has to ensure that all pallets are sorted out if one or more pistons show an incorrect piston ring assembly. The second inspection station has to make sure that no pallet passes by if the gudgeon pin is not properly secured on both sides by a guard ring.

The following defects or incorrect assemblies have to be detected by the inspection system:

- wrong type of piston ring in one or more slots;
- asymmetric ring mounted upside down;
- piston ring lacking or broken; and
- spring lacking in grooved piston ring.

Due to only slight differences or asymmetries of the rings that may be as small as 0.02 mm, high-resolution measurements are necessary to yield correct classifications.

The measurement setup consists of four cameras (one per piston) with a field of view of 18 mm height, large enough to image all three piston rings for all types of pistons. Collimated IR-solid-state lasers with microline optics project an illumination plane with 45° of inclination so as to allow for 3-D triangulation measurements of the individual ring profiles. The resulting system resolution is 0.023 mm in depth and 0.063 mm across the field of view.

Due to dedicated signal processing hardware, the system allows for simultaneous 3-D profile reconstuctions in video real-time for up to five camera/laser combinations. Thus, the measurements can be done in parallel for all four pistons of one engine while the pistons are rotated around their axis.

Once a pallet is in position, a gantry loader with multiple gripper lifts the pistons into the measuring position. It then rotates the pistons about 450° within 3 s before putting them back on the pallet. Acceleration and deceleration ramps are discarded from the measurement, so 360° are checked within 2 s, giving 100 profile scans with constant spacing.

During data acquisition, all binarization thresholds as well as the laser intensities are adapted individually for all four pistons, using hierarchical feedback control. Data acquisition and binarization is not a trivial task because scene dynamics is extreme, tapered compression rings are mat black, and grooved piston rings are chrome-plated.

Once the individual 3-D scan is found to be valid, the ring motion due to imperfect gripping ($\sigma = 0.6$ mm) and ring eccentricity ($\sigma = 2$ mm) is reconstructed. Then the assembly can be checked for the a. m. defects, except for slightly asymmetric rings mounted upside down.

The following incorrect rod assemblies have to be detected by the inspection system:

- guard ring lacking;
- guard ring mounted but not within allowances; and
- offset gudgeon pin (180° reverse mounted piston).

The assembly is checked in standstill by a the system for all eight guard rings (two per piston/rod assembly). The measurement setup comprises eight 3-D sensor units, each consisting of a matrix camera and three line-projecting solid state IR lasers. The lasers are mounted at 120° to each other, the light planes intersecting the piston and ring surface in such a way that the laser spots are almost uniformly arranged on the guard ring perimeter. By evaluating the 3-D distance of the local guard ring position at each intersection point from the taught optimum position, taking into account all mechanical tolerances of the gripping system, incorrect assemblies can be detected reliably.

Both inspection stations are fully integrated into the manufacturing line. All types of pistons assembled can be checked without any mechanical changes to the measurement setup. The robust in-line solution of these quite differing inspection tasks underline the flexibility and versatility of the light-stripe concept.

21.6 References

[1] Lenz, R. K. and Tzai, R. Y., (1987). Techniques for calibration of scale factor and image center for high accuracy 3D machine vision metrology. In *Proc. of the IEEE Intern. Conf. on Robotics and Automation*, pp. 68–75.

[2] Häusler, G., Doorsch, R., and Herrmann, J., (1994). Lasertriangulation: fundamental uncertainty in distance measurement. *Applied Optics*, **33(7)**: 1306–1314.

[3] Pfeifer, T., (1994). *Laser metrology for precision measurement and inspection in industry*, Vol. 1118 of *VDI Berichte*. Düsseldorf: VDI Verlag.

[4] Reticon, E., (1994). Solid State Camera Product. Product Information.

[5] Bresenham, J., (1996). Pixel-Processing Fundamentals. *IEEE Computer Graphics and Applications*, **16**:74–82.

[6] Ellson, R. N. and Nurre, J. H. (eds.), (1997). *Three-dimensional image capture*, Vol. 3023 of *Conf. Proc. of the SPIE*, Bellingham, WA. SPIE.

[7] (1997). *Recent advances in 3-D digital imaging and modelling*, Conference Proceedings, IEEE, May 25, 1997. Los Alamitos, CA: IEEE Computer Society Press.

[8] Ellson, R. N. and Nurre, J. H. (eds.), (1998). *Three-dimensional image capture and applications*, Vol. 3313 of *Conf. Proc. of SPIE*. Bellingham, WA: SPIE.

[9] Wan, J., Sharma, R., and Huang, T., (1998). Analysis of uncertainty bounds due to quantization for three-dimensional position estimation using multiple cameras. *Optical Engineering*, **37**:280–292.

22 A Model-Driven Three-Dimensional Image-Interpretation System Applied to Person Detection in Video Images

Bernd Radig, Olaf Munkelt, Christof Ridder,
David Hansel, and Walter Hafner

Bayerisches Forschungszentrum für Wissensbasierte Systeme (FORWISS),
München, Germany

22.1 Introduction

Automatic obtaining and describing human body motion and postures in digital video images is still a challenging field. Applications are known in a wide-ranging field from motion analysis in sports or medicine tasks up to man-machine interfaces in virtual reality tasks. This chapter presents a *model-driven system* for *detecting persons* and their 3-D posture in image sequences.

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

Early approaches for detecting postures of human bodies in video images are known from Akita [1], Hogg [2], O'Rourke and Badler [3]. These approaches have already used object models. These models have a hierarchical structure and were composed from primitives that correspond to the main parts of the human body.

This early work can be seen as the basics of newer applications that are not only in the area of surveillance and monitoring. Interacting with a virtual environment [4], serving as part of an intelligent man-machine interface, understanding the movements of persons [5, 6, 7, 8], gesture recognition [9], and ergonomics are also promising fields of recent research [10, 11, 12, 13]. Related work in tracking persons is made also in the field of motion analysis (see, for example, Section 15).

The presented system uses a strictly model-driven approach, that is, the detection of image features is initiated by a model. This model enfolds a description of the internal structure of an object, the geometric outlook of an object and, furthermore, sets of image operations that are applied to an image in order to extract relevant features. The proposed system is an image-interpretation system in the sense that it interprets parts of the image having a well-established correspondence between 3-D model features and 3-D scene features.

The system works in the 3-D space, that is, the setup of the camera(s) has to be calibrated in advance. The applied calibration technique uses a model of a pinhole camera and an approximation method that minimizes distances between the projected midpoints of a calibration table and the corresponding 2-D image points in the video images [14, 15]; see also Section 23.2.1.

The following sections present first the model, followed by a section describing appearance and its representation. The matching process is described in Section 22.4, Section 22.5 explains the implementation, and the chapter closes with sample applications.

22.2 The object model

The used model serves as a description of what can be detected in an image. This "what" is considered as an object model in contrast to a model describing, for example, the setup of the space observed by a video camera, which may have another representation. An *object model* is decomposed into parts. Thus, the object model can be considered a composition of *object model parts*, which have a well-defined inner structure and an associated geometric description. In general, this model can be applied to articulated objects. These objects should be composed of parts that correspond to the *object model parts*. Here, the model is applied to the human body.

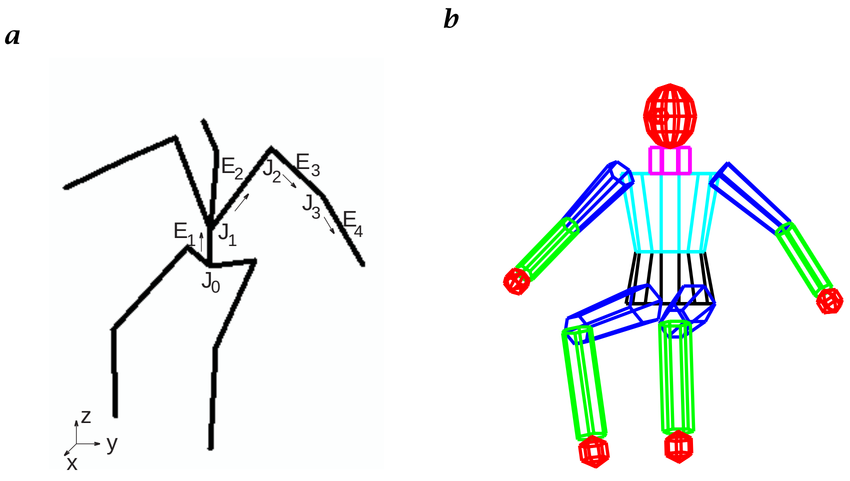


Figure 22.1:* a *The internal model;* b *the geometric model.

22.2.1 Internal model

The internal model of the object model represents the inner structure of an object. The internal model is a directed noncyclic graph

$$\begin{aligned} &G = (\mathcal{J}, \mathcal{E}) \\ &\text{with} \\ &\mathcal{J} = \{J_0, J_1, \ldots, J_n\}, \\ &\mathcal{E} = \{E_1, J_2, \ldots, E_n\} \end{aligned} \tag{22.1}$$

with J as joints and E as edges between the joints. Each pair (J_j, E_i) defines a part of the object model.

Figure 22.1(a) shows the internal model of a person. The graph corresponds to the bone structure of a person except edge E_2 and the corresponding symmetric edge. The internal model was designed for the task of tracking a person and of modeling the movements of a person appropriately. Joint J_0 can be considered as the root of the graph model.

22.2.2 Geometric model

The geometric model extends the internal model by a geometric description. Every joint $J_j \in \mathcal{J}$ has an associated 3-D point $P(J_j^s)$ and every edge $E_i \in \mathcal{E}$ has an associated 3-D line $L(E_i)$. This can be expressed by using a state vector

$$J_j^s = \left[\alpha_{J_j}, \beta_{J_j}, \gamma_{J_j}, t_{xJ_j}, t_{yJ_j}, t_{zJ_j}\right]^T \tag{22.2}$$

The state vector J_j^s describes a transformation between *object model parts*. Therefore, every object model part has a local coordinate system attached. The transformations are given by the three rotation angles $\alpha_{J_j}, \beta_{J_j}, \gamma_{J_j}$ and the translation vector $\left[t_{xJ_j}, t_{yJ_j}, t_{zJ_j}\right]^T$. Therefore, the origin of the object model parts is given by the 3-D points $P(J_j^s)$.

If one joint of the internal model serves as root joint of the object model, the 3-D point $P(\cdot)$ of connected joints can be easily obtained by

$$P(J_j^s) := T(J_{j-1}^s)T(J_{j-2}^s)\cdots T(J_0^s)\cdot P(J_0^s) \tag{22.3}$$

where $T(\cdot) \in R^{4x4}$ denotes a homogeneous transformation matrix defined by J_j^s, and J_0^s corresponds to the position of the root object model part relative to a world coordinate system. With the homogeneous transformation matrix, a 3-D rotation and a 3-D translation is described. Note that the ordering of the matrix multiplication in Eq. (22.3) is determined by the direction in the internal model according to Eq. (22.1). This means a traversing through the graph from the used object model to its predecessors, until the root object model part is reached. By this, the 3-D relation between all *object model parts* is described.

The 3-D line $L(\cdot)$ associated with an edge $E_i \in \mathcal{E}$ is defined by

$$L(E_i) := s_j \cdot \left[t_{xJ_j}, t_{yJ_j}, t_{zJ_j}\right]^T \tag{22.4}$$

with $s_j \in \mathcal{R}$ as a scaling coefficient, J_i is the corresponding joint to edge E_i, and the joint J_j is the predecessor of J_i.

The geometric model itself is built by associated geometric primitives that are defined in the local coordinate systems. The model uses for every *object model part* one rotational symmetric geometric primitive such as:

- sphere and ellipsoid; and
- modified cylinder.

The 3-D lines $L(E_i)$, see in Eq. (22.4) are the rotational axis of the geometric primitives. If more than one edge is modeled for an *object model part*, the edge that corresponds to the bone structure is used as the axis of the geometric primitve. Figure 22.1(b) shows the geometric primitives associated with the graph. It can be seen that for the edge E_2 and the corresponding symmetric edge no corresponding geometric primitive is modeled. The trunk is modeled as a modified cylinder, that is, the generating border of the cylinder is a linear combination of lines instead of just one line being parallel to the rotational axis $L(E_i)$. The head is modeled as an ellipsoid and the arms and legs also are modeled as modified cylinders.

Herewith, the geometric model serves as a description of the volumina of the body of a person. This is used to determine if for a given joint configuration $J_0^s, \ldots, J_n^s$ the geometric constraints are violated, for example, determine if the object model parts intersect. The geometric model can also be used for an evaluation of the shape of an object in an image. This can be done by the projection of the 3-D model shape to a 2-D shape in the video image based on the camera calibration.

22.3 Appearance and its representation

Besides the internal and geometric model the approach proposes the modeling of the appearance of an object in the image. This is done by linking features to the object model parts in the geometric model. These features are called model features M_i. The model features have to correspond to image features, that is, features that can be detected in the image by a segmentation process. Because the matching process is performed in 3-D, the (only) restriction for a 2-D image feature is that it has to lead to a 3-D scene feature $S_i \in S$, for example, a 3-D point or a 3-D line.

For instance, one characteristic appearance of a person is skin color. This feature can be used to detect the head of a person, if the person is dressed. The head is represented in the object model as an ellipsoid. To determine the 3-D position of this feature three questions arise:

- How can the appearance of skin color be mapped to the geometric model of a head?
- How can the appearance of color under varying illuminations be described?
- How can the 3-D position of the associated 3-D point be estimated?

22.3.1 Mapping features

The model features M_i are mapped to an object model part by a state vector $F_i(J_j) = (\alpha_{J_j}^{F_i}, \beta_{J_j}^{F_i}, \gamma_{J_j}^{F_i}, tx_{J_j}^{F_i}, ty_{J_j}^{F_i}, tz_{J_j}^{F_i})$ given in advance. Each object model part can be mapped with an arbitrary number of features. For testing purposes and in a laboratory environment it is useful to mark the joints of the human body by one landmark that corresponds to a model feature. For this purpose round plates or colored stripes can be used (see figures in Section 22.4).

By returning to the example of the head, here, the model feature of a skin-colored 3-D ellipse is used. The vector $F_1(J_j)$ for this specific feature defines the origin of the 3-D ellipse. This ellipse lies in a plane cutting the ellipsoid, which is orthogonal to the bottom and in parallel to the trunk of the object model. The corresponding 3-D scene feature S_i for the 3-D ellipse is obtained by using the given 3-D major and minor axis of the ellipse, the calculated 2-D major and minor axis of the extracted 2-D image ellipse, and the camera model used for the 3-D-2-D-projection.

This leads to possible correspondences between 3-D scene features S_i and a 3-D model feature M_i. The subsequent matching process is described in Section 22.4.

22.3.2 Color as a feature

Color is a key feature for detecting persons in images [4, 16]. However, the first step of using color information, in general, is to find an appropriate color space for the representation of uniform colors, that is, skin color. The chosen color representation has to fulfill several restrictions:

- color clusters should be as compact and well separable as possible;
- luminance information should be well separated from color information;
- the information should be coded in as few channels as possible; and
- there should be no singularities in the transformed color space.

Ohta et al. [17] used a standard technique of pattern matching theory (principal component analysis (PCA)) to compute color representations that satisfy the first restriction. Despite the fact that the "ideal" color representation with respect to color segmentation depends on the image, they found that the PCA yielded similar results for most real images. From these results they derived the $I_1 I_2 I_3$ color space. The transformation from RGB is as follows:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} (R+G+B)/3 \\ (R-B)/2 \\ (2G-R-B)/4 \end{bmatrix} \tag{22.5}$$

As a further benefit this color space codes all luminance information in channel I_1, thus satisfying the first restriction. Because color segmentation should be independent of luminance, the $I_1 I_2 I_3$ color space satisfies the third restriction, that is, channel I_1 is of no further interest for the segmentation process. Unlike the widely used *hue saturation value* (HSV) or *hue saturation intensity* (HSI) color spaces, the $I_1 I_2 I_3$ space does not contain singularities and due to linearity the calculation in this color space is very efficient (unlike, for example, the *commission internationale de l'eclairage* (CIE) color spaces).

The I_2 and I_3 components of the color space are the basis of 2-D normal probability distributions representing the objects colors. A color class Ω_k is determined by its mean vector $\boldsymbol{\mu}_k$ in the $I_2 I_3$ space and the covariance matrix K_k of its distribution

$$p(\boldsymbol{c}|\Omega_k) = p(\boldsymbol{c}|\boldsymbol{\mu}_k;K_k) = \frac{1}{2\pi\sqrt{\det K_k}}\exp(-\frac{1}{2}(\boldsymbol{c}-\boldsymbol{\mu}_k)'K_k^{-1}(\boldsymbol{c}-\boldsymbol{\mu}_k)) \tag{22.6}$$

In the system, color segmentation is implemented as a classification process with $n+1$ normal distributed classes: n object colors classes $\Omega_k, 1 \le k \le n$ and one rejection class Ω_0. The classification process determines the membership probabilities for each pixel of the color image to each of the color classes. The decision rule $\delta(\Omega_k|\boldsymbol{c})$ to minimize the risk of an incorrect classification of color vector $\boldsymbol{c}$ is as follows:

$$p_k p(\boldsymbol{c}|\Omega_k) = \max_{k=1\ldots n} p_k p(\boldsymbol{c}|\Omega_k) \tag{22.7}$$

$$\delta(\Omega_k|\boldsymbol{c}) = \begin{cases} 1, & \text{if} \quad p_k p(\boldsymbol{c}|\Omega_k) \ge \dfrac{r_f - r_z}{r_f - r_c}\displaystyle\sum_{j=1}^{n} p_j p(\boldsymbol{c}|\Omega_j) \\ 1, & \text{otherwise} \end{cases} \tag{22.8}$$

with
r_f : costs of incorrect classification;
r_z : costs of rejection;
r_c : costs of correct classification; and
$r_c < r_z \le r_f$.

Equations (22.7) and (22.8) define a Bayes classifier to minimize the costs of a false classification.

Because of changes in illumination during the processing of image sequences, the color classification requires an adaptation of the color classes. A modified decision-supervised learning algorithm [18] is used to control the variations of the colors: a feature vector is first classified and then the classifier itself is updated using the just classified feature vector by means of supervised learning [19].

Figure 22.2(a) shows the result of the color pixel classification. By applying a morphological closing operator followed by a 2-D ellipse fitting of the resulting binary image region the 2-D major and minor axes are obtained that are needed for the calculation of a 3-D scene feature. The 2-D ellipse in Fig. 22.2(b) is the corresponding image feature of the head.

22.3.3 Estimating three-dimensional positions of features

The extraction process of 2-D image features only determines pixel positions in the image. However, the matching process needs 3-D positions. According to the used explicit camera model, a pixel position leads to a 3-D line-of-sight. To get the 3-D position of a feature,

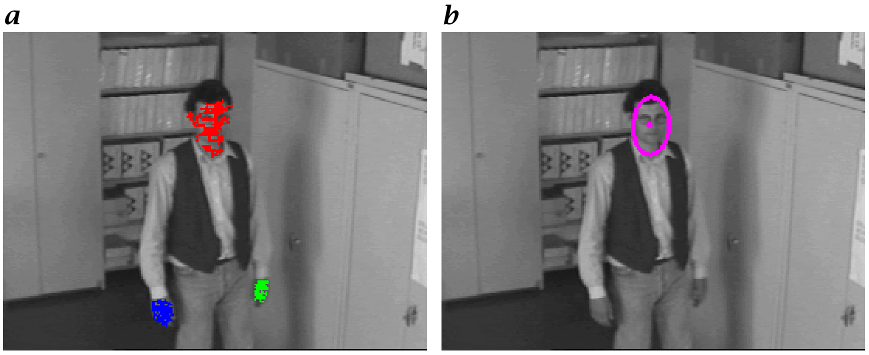


Figure 22.2: ***a*** *Applying a color classification;* ***b*** *followed by an ellipse fitting for the head feature.*

depth information also is needed. Basically, two different approaches are used to get this 3-D information: either a monocular or a stereo approach, called interfaces of the system. A set of image-processing operators is applied to these interfaces in order to obtain image features such as rectangles, boxes, circles or ellipses. The decision as to what interface should be applied depends on the application and on the necessary accuracy. Linking the model to the interface is quite simple: An image feature linked to the model has to have an appropriate segmentation routine in the used interface.

Monocular interface. To estimate the depth by the monocular approach, knowledge of the related object model part is needed. This means, for example, for the head feature that four lines-of-sight are calculated as tangents to the extremal points of the image ellipse: one line to the top, the bottom and to the left and the right of the ellipse. The estimated depth is given by the position where the elliptic sphere of the geometric model fits best into the 3-D cone given by the four lines-of-sight. To improve the depth estimation it is possible also to use the height of the head (= the component t_{zJ_j} of $P(J_j^s)$). For this, the assumption is made that the person is standing. Herewith, the depth is given where the height of the head fits best into the triangle, given by the line-of-sight at the center of the image ellipse and the xy-plane of the world coordinate system.

Stereo interface. A higher accuracy of the depth estimation can be reached while using a binocular setup to obtain 3-D information. The baseline of the two cameras is given by the calibration of the camera positions in a world coordinate system. An image feature has to be extracted in both camera images, which leads to two related 3-D lines-of-sight. The theoretical crossing of the two lines determine the cor-

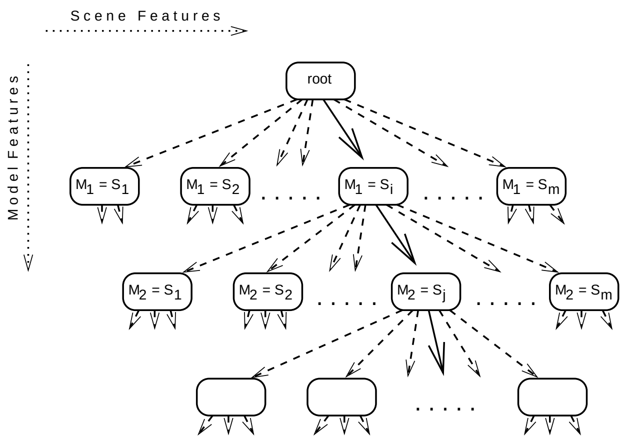


Figure 22.3: *The model-driven interpretation tree.*

responding 3-D point. The system is looking for the closest distance between the two lines-of-sight instead of real crossing [20].

22.4 Matching

The matching process is performed in the 3-D space. It is divided into two steps. The first step builds associations between 3-D scene features $S_i \in S$ and the 3-D model features M_i, which are associated with the 3-D points $P(J_j^s)$ by $F_i(J_j)$. This is done by using an interpretation tree; see Fig. 22.3 [21, 22, 23].

A "binary length constraint" is used to restrict the search in the search space that is spanned by the number of joints

$$r((S_i, S_j), (S_j, S_k), L(E_i), L(E_j)) = \text{true} \Leftrightarrow \frac{\|\overline{S_i{}' S_j{}'}\|}{\|\overline{S_j{}' S_k{}'}\|} = \frac{\|L(E_i)\|}{\|L(E_j)\|} \tag{22.9}$$

where $S_i{}'$ are the scene features S_i translated by the state vector $F_i(J_j)$; see Section 22.3.1. The "binary length constraint" takes the 3-D distances between the adjacent object model parts into account. This is supported by the rigid bone length of human bodies.

The internal model itself (see Section 22.2.1), defines the search direction and *serves* as the *interpretation tree*. Thus applying Eq. (22.9) to the situation in Fig. 22.1 leads to

$$\frac{\|\overline{S_i{}' S_j{}'}\|}{\|\overline{S_j{}' S_k{}'}\|} = \frac{\|L(E_1)\|}{\|L(E_2)\|} \tag{22.10}$$

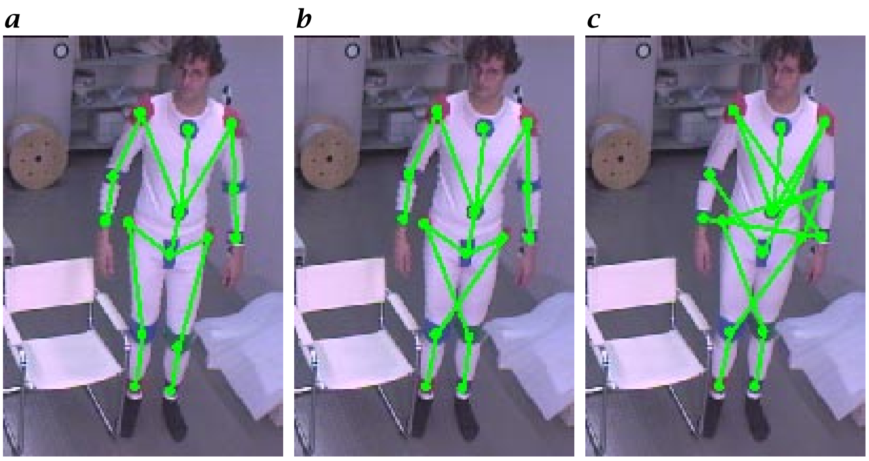


***Figure 22.4:* a** *and* **b** *Correct matching according to $\mathcal{I}$;* **c** *the result after a higher 3-D tolerance is used in the bone length restriction (each image taken from the right camera).*

on the first level of the interpretation tree, and

$$\frac{\|\overline{S_j{}'S_k{}'}\|}{\|\overline{S_k{}'S_l{}'}\|} = \frac{\|L(E_2)\|}{\|L(E_3)\|} \tag{22.11}$$

on the second level of the interpretation tree. The result of traversing the interpretation tree are valid mappings between the scene features S_i and the model features M_j representing the joints J_j. These valid mappings are called the interpretation

$$\mathcal{I} = \left\{(S_i, J_j) | S_i \in \mathcal{S}, J_j \in \mathcal{J}\right\} \tag{22.12}$$

Figure 22.4a and b show examples of a valid mapping. For demonstration and testing purposes, the stereo approach is used here. Colored landmarks are mounted close to the joints of the person. These image joints correspond to the joints of the model. Again, the appearance of the landmarks is determined by an algorithm using stereo vision to obtain the corresponding 3-D scene feature.

Figure 22.4b shows an example for a correct "false mapping": correct according to $\mathcal{I}$, and false because the feet are swapped. The behavior of exchanging symmetric parts has also been reported by Grimson and Lozano-Perez [23]. Note that this interpretation is rejected by using the restriction of the geometric model: legs cannot cross each other.

For the mapping process used in Fig. 22.4(c) a higher 3-D matching tolerance was choosen. This means, that in Eq. (22.9) the restriction $r(\cdot)$ is true even if the 3-D distances between the transformed scene features $\|\overline{S_i{}'S_j{}'}\|$ and the length of model edges $\|L(E_i)\|$ fit only approximately. The right shoulder of the person is marked by a red landmark, the right elbow is marked by a violet landmark, and the right hand is marked by a green landmark. The left hand is also marked by a violet mark. Thus, a part of another valid mapping of $\mathcal{I}$ is the combination of the 3-D points in the right shoulder, the left hand and the right hand as a matching of the right arm (shoulder, elbow, hand) of the model. It is obvious that the distance between the right shoulder (red) and the left hand (violet) is longer than the distance between the shoulder (red) and the right elbow (violet) of the correct match.

To reflect different body sizes, the object model can be adapted in size if multiple images are processed. To achieve this, the translation vector $(t_{xJ_j}, t_{yJ_j}, t_{zJ_j})$ of each object model part is adapted by using a least-square approach that determines the scaling factor s_j in Eq. (22.4). By this, the 3-D matching tolerance becomes smaller for a longer tracking process of a person.

The second step during the matching process is to determine a configuration of the state vector J_j^s that satisfies Eq. (22.12). Only the translation vector $(t_{xJ_j}, t_{yJ_j}, t_{zJ_j})$ of each object model part is given by traversing the interpretation tree, using the correspondences between 3-D scene features S_j and joints J_j. This is because the traverse uses only a length constraint. Thus, the direction, that is, $(\alpha_{J_j}, \beta_{J_j}, \gamma_{J_j})$ of the object model parts have to be determined. Each angle is further restricted by an interval defining its minimal and maximal rotation. This interval is subsampled by 5° steps. By using a backtracking algorithm starting from the endpoints of the internal model (see Eq. (22.1)), $(\alpha_{J_j}, \beta_{J_j}, \gamma_{J_j})$ can be obtained. Using this restriction of angles the mapping of Fig. 22.4(c) can now be rejected.

The applied calculation of the angles can be seen in Fig. 22.5. Figure 22.5(a,b) shows the correct mapping of the features for the right and the left camera, respectively. According to this, Fig. 22.5(c,d) shows the projected geometric model superimposed on the original image. Note that the orientations of the arms and legs are correct.

It should be pointed out that the use of the same landmarks for every joint, similar to the example in Fig. 22.5, results in more possible combinations in the interpretation tree in contrast to the use of different marks (see Fig. 22.4). With the latter, the interpretation tree becomes smaller and the matching process faster.

22.5 Implementation

The system is fully implemented by using a C++ class library in order to handle the complex data structures efficiently. The underlying image operators are provided by the image analysis tool HALCON [24]. HAL-

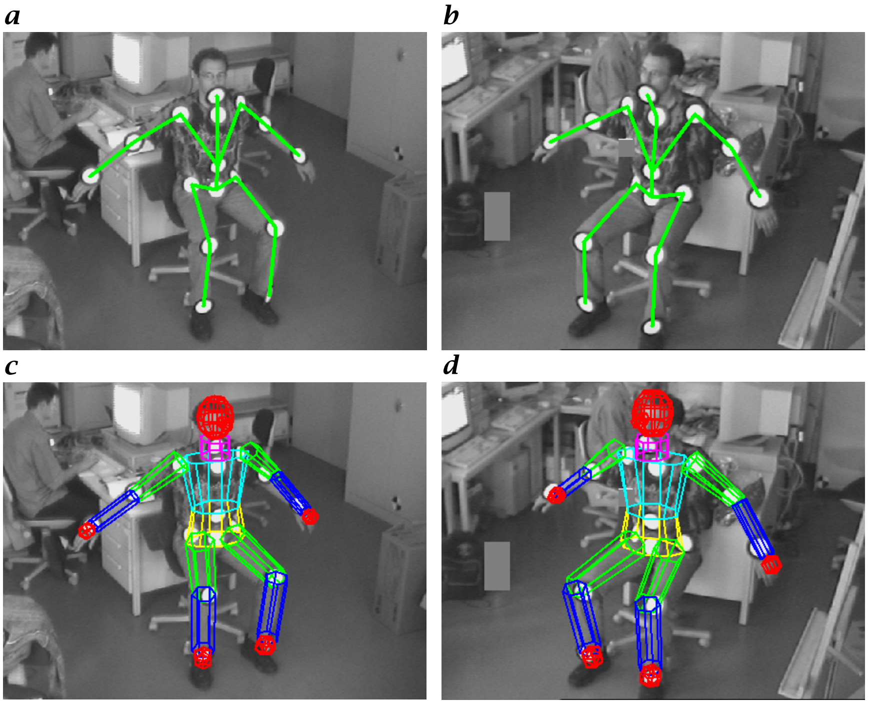


Figure 22.5: *Correct matching of the detected image features to the joints of the object model (**a** left camera; **b** right camera). According to the matching, the geometric model is superimposed on the images (**c** left camera; **d** right camera).*

CON also has a dedicated class hierarchy. Figure 22.6 shows a part of the implemented class hierarchy. An overview of software engineering and programming languages for image processing is given in Section 5.

The main class is the class <ObjectModel> that serves as a container for the image-interpretation system. The class has a class <ObjectModelPart> as a root object model part. <ObjectModelPart> reflects the internal structure of the model; see Eq. (22.1).

An <ObjectModelPart> must have at least one <Feature>. This class links the sets of image operators to the corresponding object model part by the knowledge of a model interface (see the following). Furthermore, it also performs the calculation of the 3-D scene features. Three-dimensional scene features are registered in a class variable of the class <Feature> in order to have a uniform access to the scene features during the matching process.

Specializations of the class <ObjectModelPart> are the classes <OMP_Sphere> and <OMP_ModifiedCylinder> that correspond to the geometric model of the object model. The classes <F_Ellipse>, <F_Landmark> and <F_ColoredBlob> correspond to the introduced

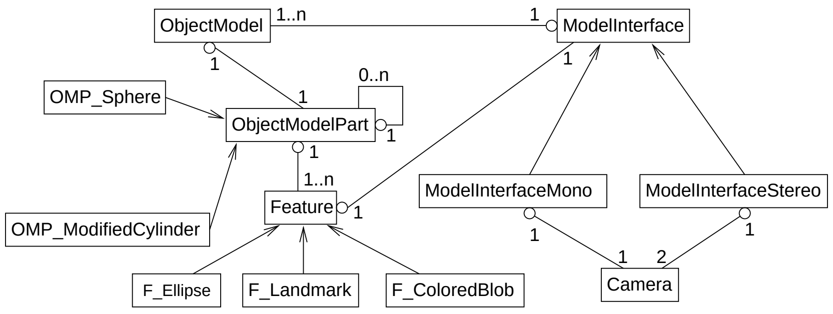


Figure 22.6: *A part of the class hierarchy of the implemented system (using UML-syntax).*

methods of obtaining 3-D scene features: <F_Ellipse> for the skin color-based matching of the head and <F_Landmark>, <F_ColoredBlob> in the stereo approach for testing purposes with landmarks.

For the transparent handling of the mono and stereo approach, the class <ModelInterface> is designed. The derived classes <ModelInterfaceMono> and <ModelInterfaceStereo> own one or two cameras. The model interfaces control the segmentation of the 2-D image features from the images and support the class <Feature> for calculating the 3-D scene features. The matching process is also done by the class <Feature>. The model interfaces know the currently found model instances and allow display of the models in the image. It should be remarked again that the whole detection process is controlled by the class <ObjectModel>. To guarantee this, the model interface activates the instances of object models after getting the next image of the sequence. An instance of a prototypic object model is always available to take the appearance of new objects in the scene into account.

With this design, it is quite easy to extend the system to more than two cameras. This is necessary for tasks where the exact positions of the joints have to be measured or where it is not possible to cope with covered image parts (e.g., landmarks) for a longer time.

The systems runs on an Ultra Sparc (167 MHz) and can process approximately 2 frames/s. The most time-consuming task is the matching process. Because the system setup during this step is designed for learning the movements of the person, there is now the need for real-time tracking.

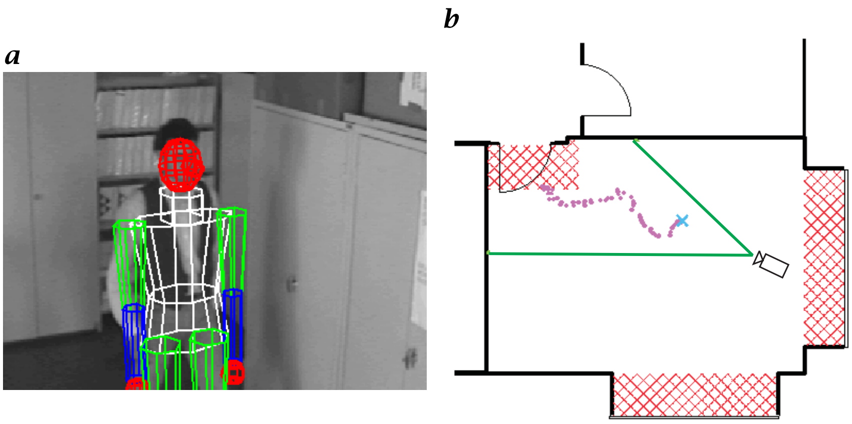


Figure 22.7: ***a*** *Detected person in the laboratory superimposed with the model;* ***b*** *floor plan of the laboratory with the marked path.*

22.6 Applications

The two different model interfaces lead to different applications. The mono interface is predetermined for *surveillance* and monitoring tasks where the position of human bodies should be estimated. Therefore, only the skin-colored ellipse feature is mapped to the head of the model. According to the example of the color classification in Fig. 22.2, in Fig. 22.7a the detected model position is presented. Here, the assumption is made that the person is walking; thus the model is used in an upstanding posture (no more features are used to determine the exact posture). Figure 22.7b shows the result of the application: in the floor plan of the laboratory the estimated 3-D positions are projected and mark the path of the detected person step by step. Alarm zones are marked in the plan of the observed room. If the path crosses one of these zones an alarm can be triggered.

The 3-D position can also be used to control pan/tilt cameras for monitoring tasks. The cameras follow the person if he/she leaves the current image area. By combining several calibrated pan/tilt cameras, the 3-D information on a tracked person can be handed over from one camera to another. Herewith, the monitoring of persons in wide areas is possible.

An approach of the stereo configuration is the estimation of the 3-D positions of all joints. These sets of positions can be used by *virtual reality* systems or for determining configurations of computer-aided design (CAD) models. These models are used, for example, in the automobile industry for analysis in *ergonomics*. Figure 22.8 shows four images out of a sequence of 100 images. The sequence shows a person

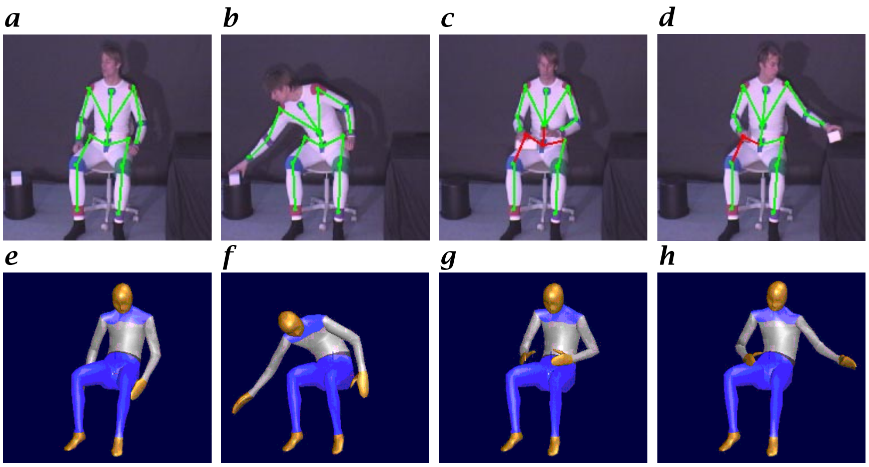


Figure 22.8: ***a–d*** *Four images out of a sequence of the right camera;* ***e–h*** *corresponding postures of the animated CAD-model RAMSIS.*

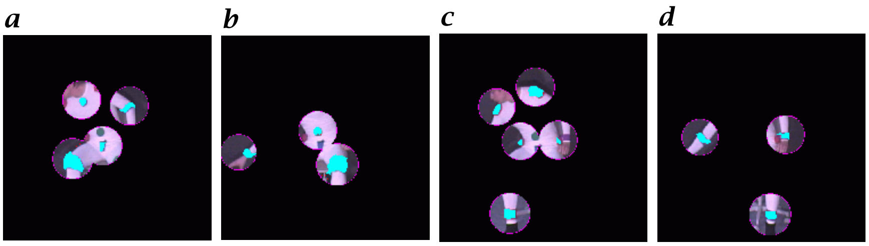


Figure 22.9: *The projected search areas corresponding to images in Fig. 22.8 (**a** blue, **b** green, **c** red, **d** violet).*

who takes a white box with his right hand, transfers the box to his left-hand and puts it on a black box on his left hand side. In Fig. 22.8a-d the matching of the determined joint configuration to the 2-D image features is shown (all images from the right camera). According to this the images in Fig. 22.8e-h show the animated CAD model.

When the person moves the box in front of the body two of the landmarks are covered. This can be seen in Fig. 22.8c. In this case no 3-D positions can be estimated for the corresponding scene feature, for example, to the right hip. However, the mapping process uses interpolated positions until the features can be segmented again.

To speed up the matching process, the number of found scene features should be kept small. Therefore, the region of interest in the images has to be reduced. A segmentation of the images in foreground and background regions (see [25]) is a possible approach. However, the use of the interpolated positions as predicted positions of the features for the next image also is useful. The predicted positions and the direction of the movements of the joints determine a 3-D search space. A projection of these 3-D spaces leads to the regions of interest in the images based on the used calibration. Figure 22.9 shows the search regions for the blue (a), the green (b), the red (c), and the violet landmark (d) according to the image (Fig. 22.8b).

Furthermore, the number of possible combinations in the interpretation tree of the matching process is reduced by the prediction of the search spaces. This is because the matching process allows only 3-D scene features in the specific 3-D search spaces. In comparison with the example in Fig. 22.4c, this means that the right elbow (marked violet) can not be taken for the left hand (also marked violet) any longer after a correct matching of the right arm.

The prediction of the positions can be improved by learning the movements of a person by means of extracting appropriate features. One way is to construct feature vectors using $(\alpha_{J_j}, \beta_{J_j}, \gamma_{J_j})$ over time. The aim of further development is to identify a periodic motion in order to describe a motion in a classification framework.

Acknowledgment

The authors would like to thank their colleague Christoph Zierl for fruitful discussions and critical remarks on the design of the system. Thanks also to Sónia Antunes, Thomas Diederich, Felix Ko, and Ingmar Rauschert. The application that combines the system with the CAD model RAMSIS is the result of the cooperation with the institute of ergonomics (Lehrstuhl für Ergonomie) at the Technische Universität München.

22.7 References

[1] Akita, K., (1984). Image sequence analysis of real world human motion. *Pattern Recognition*, **17**(1):73–83.

[2] Hogg, D., (1983). Model-based vision: a program to see a walking person. In *Image and Vision Computing*, Vol. 1(1), pp. 5–20. Amsterdam, NL: Elsevier Science.

[3] O'Rourke, J. and Badler, N. I., (1980). Model-based image analysis of human motion using constraint propagation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **2**(6):522–535.

[4] Wren, C., Azarbayejani, A., Darrell, T., and Pentland, A., (1995). *Pfinder: Real-Time Tracking of the Human Body*. Technical Report 353, Massachusetts Institute of Technology, M.I.T. Media Laboratory, Perceptual Computing Section.

[5] Bregler, C., (1997). Learning and recognizing human dynamics in video sequences. In *Computer Vision and Pattern Recognition*, pp. 568–574. Washington, DC: IEEE Computer Society Press.

[6] Leung, M. K. and Yang, Y.-H., (1987). Human body motion segmentation in a complexe scene. *Pattern Recognition*, **20**:55–64.

[7] Leung, M. K. and Yang, Y.-H., (1994). *An Empirical Approach to Human Body Motion Analysis*. Technical Report 94-1, University of Saskatchewan. Canada.

[8] Rohr, K., (1994). Towards model-based recognition of human movements in image sequences. *CVGIP: Image Understanding*, **59**(1):94–115.

[9] Wilson, A. D. and Bobick, A. F., (1997). *Recognition and Interpretation of Parametric Gesture*. Technical Report 421, Massachusetts Institute of Technology, M.I.T. Media Laboratory, Perceptual Computing Section.

[10] Chen, Z. and Lee, H.-J., (1992). Knowledge-guided visual perception of 3D human. *IEEE Trans. Systems, Man, and Cybernetics*, **22**(2):336–342.

[11] Daucher, N., Dhome, M., Lapreste, J., and Rives, G., (1993). Modelled object pose estimation and tracking by monocular vision. In *British Machine Vision Conference*, Vol. 1, pp. 249–258.

[12] Leung, M. K. and Yang, Y.-H., (1995). First sight: a human body outline labeling system. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **17**(4):359–377.

[13] Munkelt, O. and Kirchner, H., (1996). STABIL: A system for monitoring persons in image sequences. In *Image and Video Processing IV*, Vol. 2666 of SPIE Proceedings, pp. 163–179. SPIE—The Intern. Soc. for Optical Engineering.

[14] Lanser, S., Zierl, C., and Beutlhauser, R., (1995). Multibildkalibrierung einer CCD-Kamera. In *Mustererkennung*, G. Sagerer, S. Posch, and F. Kummert, eds., Informatik aktuell, pp. 481–491, Deutsche Arbeitsgemeinschaft für Mustererkennung. Berlin: Springer.

[15] Lenz, R. and Tsai, R. Y., (1988). Techniques for calibration of the scale factor and image center for high accuracy 3D Machines Metrology. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **10**(5):713–720.

[16] Schuster, R., (1994). Adaptive modeling in color image sequences. In *Mustererkennung*, Informatik Xpress 5, pp. 161–169. Deutsche Arbeitsgemeinschaft für Mustererkennung.

[17] Ohta, Y.-I., Kanade, T., and Sakai, T., (1980). Color information for region segmentation. *Computer Graphics and Image Processing*, **13**:222–241.

[18] Niemann, H., (1989). *Pattern Analysis and Understanding*, Second edition, chapter 4. Springer Series in Information Sciences. Berlin: Springer.

[19] Hafner, W. and Munkelt, O., (1996). Using color for detecting persons in image sequences. *Pattern Recognition and Image Analysis*, **7**(1):47–52.

[20] Xu, G. and Zhang, Z., (1996). *Epipolar Geometriy in Stereo, Motion and Object Recognition*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

[21] Grimson, W. E. L., (1989). The combinatorics of object recognition in cluttered environments using constrained search. *Artificial Intelligence*, **11**(6):632–643.

[22] Grimson, W. E. L., (1989). On the recognition of curved objects. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **11**(6):632–643.

[23] Grimson, W. E. L. and Lozano-Perez, T., (1984). Model-based recognition and localization from sparse range or tactile data. *Int. J. Robotics Research*, **3**(3):3–35.

[24] Eckstein, W. and Steger, C., (1996). Interactive data inspection and program development for computer vision. In *Visual Data Exploration and Analysis III*, Vol. 2656 of SPIE Proceedings, pp. 296–309. SPIE—The Intern. Soc. for Optical Engineering.

[25] Ridder, C., Munkelt, O., and Kirchner, H., (1995). Adaptive background estimation and foreground detection using kalman-filtering. In *Proc. International Conference on recent Advances in Mechatronics, ICRAM'95*, O. Kaynak, M. Özkan, N. Bekiroğlu, and İ. Tunay, eds., pp. 193–199, UNESCO Chair on Mechatronics. Boğaziçi University, 80815 Bebek, Istanbul, TURKEY.

23 Model-based Three-dimensional Object Recognition from Single-Perspective Views

Stefan Lanser, Christoph Zierl, and Bernd Radig

Forschungsgruppe Bildverstehen (FG BV), Informatik IX
Technische Universität München, München, Germany

23.1 Introduction

One of the fundamental requirements for an *autonomous mobile system* (AMS) is the ability to navigate within an *a priori* known environment and to recognize task-specific objects, that is, to identify these objects and to compute their 3-D pose relative to the AMS. For the accomplishment of these tasks the AMS has to survey its environment by using appropriate sensors. This contribution presents the vision-based *3-D object recognition* system MORAL (*M*unich *O*bject *R*ecognition *a*nd *L*ocalization, http://wwwradig.in.tum.de/projects/moral.html), which performs a model-based interpretation of single video images of a charge coupled device (CCD) camera. Using appropriate parameters, the system can be adapted dynamically to different tasks. The communication with the AMS is realized transparently using *remote procedure*

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

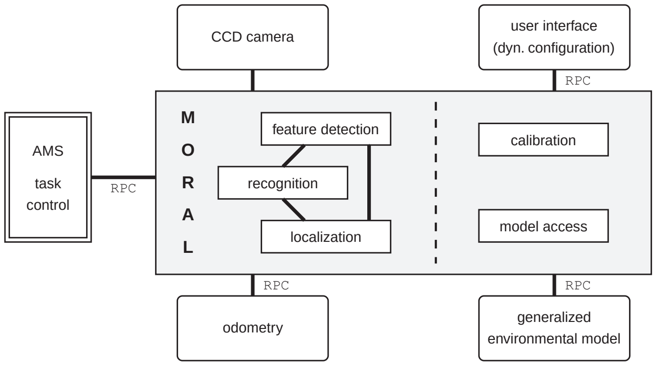


Figure 23.1: *Overview of MORAL.*

calls. As a whole this architecture enables a high level of flexibility with regard to the used hardware (computer, camera) as well as to the objects to be recognized.

In the context of autonomous mobile systems the following tasks can be solved using a vision-based object recognition system:

- recognition of task-specific objects (e.g., doors, trashcans, or workpieces);
- localization of objects (estimation of the 3-D pose relative to the AMS) in order to support manipulation tasks; and
- navigation in an *a priori* known environment (estimation of the 3-D pose of the AMS in the world).

These tasks can be formalized by the interpretation

$$\mathcal{I} \;=\; \langle\; obj, \left\{(I_{j_1}, M_{i_1}), \ldots, (I_{j_k}, M_{i_k})\right\}, (\boldsymbol{R}, \boldsymbol{t}) \;\rangle \tag{23.1}$$

with the object hypothesis obj, the correspondence (I_{j_l}, M_{i_l}) between image feature I_{j_l} and the model feature M_{i_l}, and $(\boldsymbol{R}, \boldsymbol{t})$ the estimated 3-D pose of the object.

The object-recognition system MORAL presented in this chapter accomplishes these tasks by comparing the predicted model features with the features extracted from a video image [1, 2]. The underlying 3-D models are polyhedral approximations of the environment provided by a hierarchical world model. This contribution deals mainly with rigid objects. The extension of this work to the recognition of articulated objects, that is, objects consisting of multiple rigid components connected by joints, is shown briefly in Section 23.2.6.

Related work to the recognition task can be found in Dickinson et al. [3], Grimson [4], Ikeuchi and Kanade [5], and Pope [6]. The problem of the self-localization of an AMS is considered in Fennema et al. [7], Christensen and Kirkeby [8], and Kosaka and Pan [9]. Common to both tasks are the use of a geometric model and the basic localization procedure, that is, the determination of the 6 degrees of freedom (DOF) pose of the AMS relative to the world or to an object, respectively.

23.2 The MORAL object-recognition system

The presented object-recognition system (see Fig. 23.1) is implemented as a *remote procedure call* (RPC) server, which is called by an arbitrary client process, especially by the AMS task control system. The standardized RPC mechanism allows a hardware independent use of the MORAL system. Therefore, MORAL can easily be applied on different platforms. By using the same mechanism, MORAL communicates (optionally) with other components, for example, the *generalized environmental model* (GEM). With the help of specific RPCs MORAL can be dynamically configured, and can thus be flexibly adapted to modified tasks. The internal structure of MORAL consists essentially of the following five modules, which are implemented in ANSI-C and C++, respectively:

- The *calibration* determines off-line the internal parameters of the used CCD camera as well as the relative pose of the camera with respect to the AMS (*hand-eye calibration*).
- The *model prediction* generates 2-D views suitable for the recognition. This is performed by appropriate requests to the *generalized environmental model*.
- The *feature detection* extracts online the necessary features from the video image.
- The *recognition* generates online hypotheses of objects in the view-field of the CCD camera and their rough 3-D pose.
- Using a starting hypothesis, the *localization* determines the exact 3-D pose (6 DOF) of the object relative to the AMS or the exact 3-D pose of the AMS in the world, respectively.

These modules are described in the following sections in more detail.

23.2.1 Calibration

In order to obtain the 3-D object pose from the grabbed video image, the internal camera parameters (mapping the 3-D world into pixels) as well as the external camera parameters (pose of the CCD camera relative to the manipulator or the vehicle) have to be determined with sufficient accuracy.

Internal camera parameters. The proposed approach uses the model of a pinhole camera with radial distortions to map 3-D point in the scene into 2-D pixels of the video image [10], see Section 17.4. It includes

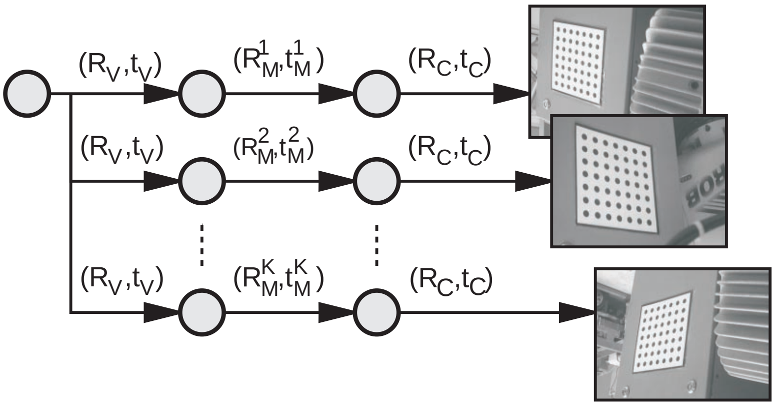


Figure 23.2: *Estimation of the camera pose based on known relative movements of the robot manipulator.*

the internal parameters as well as the external parameters $\boldsymbol{R}$, a matrix describing the orientation, and $\boldsymbol{t}$, a vector describing the position of the camera in the world.

In the first stage of the calibration process the internal camera parameters are computed by simultaneously evaluating images showing a 2-D calibration table with N circular marks P_i taken from K different viewpoints; see Section 17.5.2. This *multiview calibration* [11] minimizes the distances between the projected 3-D midpoints of the marks and the corresponding 2-D points in the video images. The 3-D pose $(\boldsymbol{R}, \boldsymbol{t})$ of the camera is estimated during the minimization process. Thus, only the model of the calibration table itself has to be known *a priori*.

Hand-eye calibration. Once the internal camera parameters have been determined, the 3-D pose of the camera relative to the *tool center point* is estimated in the second stage of the calibration process (*hand-eye calibration*).

In the case of a camera mounted on the manipulator of a mobile robot the 3-D pose of the camera $(\boldsymbol{R}, \boldsymbol{t})$ is the composition of the pose of the robot $(\boldsymbol{R}_V, \boldsymbol{t}_V)$, the relative pose of the manipulator $(\boldsymbol{R}_M, \boldsymbol{t}_M)$, and the relative pose of the camera $(\boldsymbol{R}_C, \boldsymbol{t}_C)$; see Fig. 23.2. The unknown pose $(\boldsymbol{R}_C, \boldsymbol{t}_C)$ is determined by performing controlled movements $(\boldsymbol{R}_M^k, \boldsymbol{t}_M^k)$ of the manipulator similar to [12]

$$e(\boldsymbol{x}) = \sum_{k=1}^{K} \sum_{i=1}^{N} \| \tilde{\boldsymbol{s}}_i^k \times c_i(P_i, \boldsymbol{x}^k) \|^2 \longrightarrow \min \tag{23.2}$$

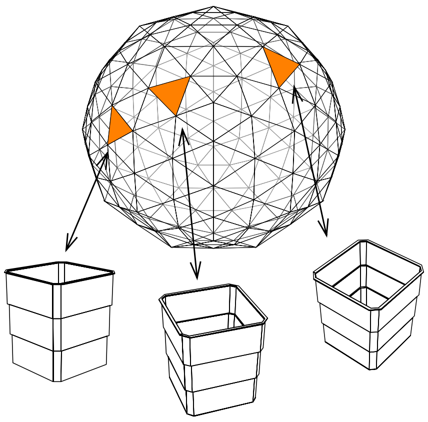


Figure 23.3: *Each triangle of the tessellated Gaussian sphere defines a 2-D view of an object.*

with $\tilde{\boldsymbol{s}}_i^k$ the normalized vector of the line-of-sight through the extracted 2-D point $\tilde{p}_i^k$ in the k^{th} video image and $c_i(P_i, \boldsymbol{x}^k)$ the 3-D position of a mark on the calibration table transformed to the camera coordinate system. The vector $\boldsymbol{x}^k$ comprises the internal and external camera parameters. The minimization process determines the unknown parameters $(\boldsymbol{R}_V, \boldsymbol{t}_V)$ and $(\boldsymbol{R}_C, \boldsymbol{t}_C)$ [11].

Because the used 2-D calibration table is mounted on the mobile robot itself, the manipulator can move to the different viewpoints for the *multiview calibration* automatically. Thus, the calibration can be accomplished in only a few minutes.

23.2.2 Model prediction

The recognition of rigid objects is based on a set of characteristic 2-D views (*multiview representation*). Here, in contrast to Section 8.3.2, a fixed set of 320 perspective 2-D views is determined using the tessellated Gaussian sphere. These views contain the model features used for establishing the correspondences (I_{j_l}, M_{i_l}) part of $\mathcal{I}$ in Eq. (23.1). This model prediction is provided by the GEM [13] based on the boundary representation of a polyhedral 3-D object, which can be derived from a computer-aided design (CAD) model. The tessellated Gaussian sphere and three 2-D views of the object TRASHCAN are shown in Fig. 23.3.

Referring to Eq. (23.1), obj is not limited to a specific object. In the case of vision-based navigation it means the environment of the AMS. Instead of using a set of characteristic 2-D views, the expected 2-D model features are provided by GEM according to the *expected position* of the AMS.

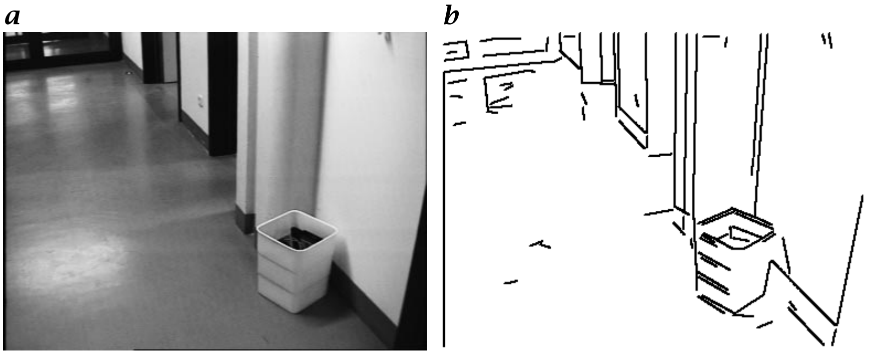


Figure 23.4: ***a*** *Original video image;* ***b*** *detected image features.*

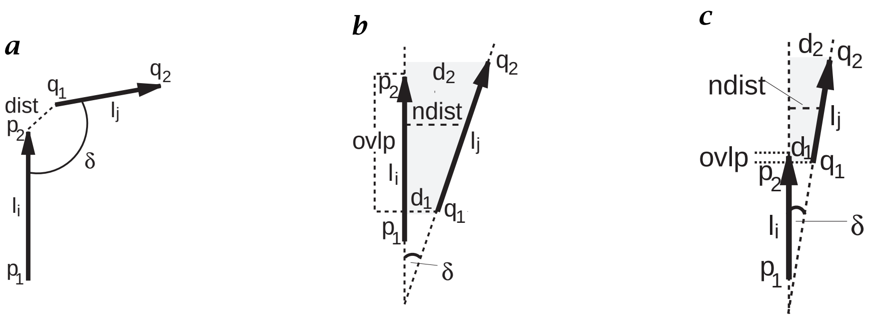


Figure 23.5: *Local binary constraints used:* ***a*** $\langle l_i$ *Vertex* $l_j\rangle$; ***b*** $\langle l_i$ *OvlPar* $l_j\rangle$; ***c*** $\langle l_i$ *Cont* $l_j\rangle$.

23.2.3 Feature detection

The detection of image features is based on the image analysis system *HALCON* [14] (`http://www.MVTec.com/halcon.html`, formerly HORUS). HALCON provides a large number of different image operators that are controlled by MORAL according to the performed task. Object-specific knowledge is used to control the image-segmentation process. This knowledge comprises different types, for example, lines, faces, arcs, and features such as color of the foreground or background (Fig. 23.4). The feature-detection module can be parameterized either by the modules of MORAL itself or by the clients that call services of MORAL via the RPC mechanism, for example, the AMS task control or the user interface.

The interpretation is based mainly on straight line segments, which are derived from a gradient image (see Section 10.3) by means of thresholding, pixel linkage, and a polygon approximation of the detected image contours. These line segments are grouped by the three local binary constraints [15]: *Vertex* (two line segments form a vertex); *OvlPar* (two

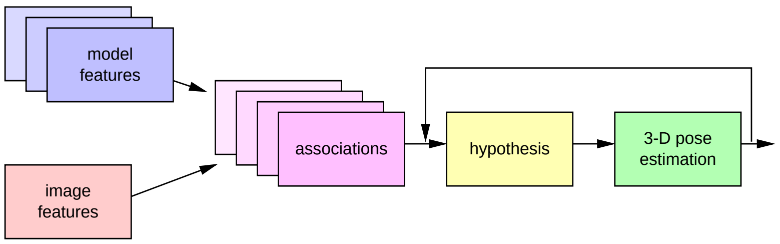


Figure 23.6: *Overview of the recognition process.*

line segments are nearly parallel and do overlap); and *Cont* (one line segment is the continuation of another one); see Fig. 23.5. The *Cont* relation is used to compensate for broken image line segments. The other two relations *Vertex* and *OvlPar* are used for the matching process, see Sections 23.2.4 and 23.2.5. Vertices and parallel line segments are further classified by local criteria to determine which constraints are *compatible* or *inconsistent* [16].

23.2.4 Object recognition

The aim of the object-recognition module is to identify objects and to determine their rough 3-D pose by searching for the appropriate 2-D model view matching the image. This is done by establishing correspondences between image features extracted from the CCD image and 2-D model features of an object (see Fig. 23.6). Obviously, if a rough 3-D pose is *a priori* known, the recognition module is "bypassed" and the localization module is called directly (e.g., continuous self-localization).

Building associations. The first step in the object-recognition process is to build a set of *associations*. An association is defined as a quadruple (I_j, M_i, v, c_a), where I_j is an image feature, M_i is a model feature, v is one of the characteristic 2-D model views of an object, and c_a is a confidence value of the correspondence between I_j and M_i. This value is obtained by traversing aspect-trees [17] or by a geometrical comparison of the features incorporating constraints based on the topological relations *Vertex*, *OvlPar*, and *Cont*; see Section 23.2.3. The latter measures are taken to determine to which extent the model constraints can be fulfilled in the image. Associations resulting in inconsistent constraints are discarded.

Building hypotheses. In order to select the "correct" view of an object the associations are used to build *hypotheses* $\{(obj, \mathcal{A}_i, v_i, c_i)\}$. For each 2-D view v_i all corresponding associations with sufficient confidence are considered. From this set of associations the subset of associations $\mathcal{A}_i$ with the highest rating forming a *consistent labeling*

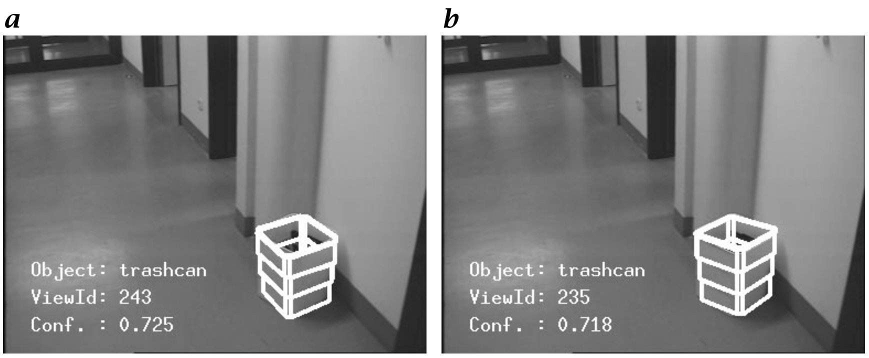


Figure 23.7: *The two highest ranked hypotheses according to the image features in Fig. 23.4 and the object model in Fig. 23.3.*

of image features is selected. Specifically, the geometric transformations in the image plane between model and image features in $\mathcal{A}_i$ are similar, that is, they form a cluster in the transformation space. Furthermore, the image features must be subject to the same topological constraints as their corresponding model features [15]. The confidence value c_i depends on the confidence values of the included associations and the percentage of mapped model features. The result of the described recognition process is a ranked list of possible hypotheses (see Fig. 23.7), which are verified and refined by the localization module.

A view in a hypothesis determines one translational and two rotational degrees of freedom (DOF) of the object pose. Additionally, while building the hypotheses a scale factor and a translational vector in the image plane is computed. Using this weak perspective, a rough 3-D position of the object is derived. Note that some of the hypotheses from the object-recognition module may be "incorrect." These hypotheses are rejected by the subsequent verification and refinement procedure performed by the localization module.

23.2.5 Localization

The aim of the localization module YALE (Yet Another Location Estimator) [16] is the determination of all 6 DOF of the CCD camera relative to an object or to the environment. The input for this module is a rough 3-D pose and a set of 3-D model lines. This input can be either a model of the environment (provided by GEM) or of a specific object (result of the previous object recognition). The module will be considered either with an image, with the order to grab an image online, or with given image lines. In the latter case, fixed correspondences can be assigned. Otherwise, image lines are extracted online from the image.

If the uncertainties of the pose and/or the position of the model lines are known, specific search spaces for the extraction of image lines are computed [18]. Final correspondences are established using a model-driven *interpretation-tree* approach [4]. Traversing the tree is controlled by the *viewpoint consistency constraint* (based on the remaining error after the pose refinement) and the maximum topological support. A penalty term for *not-in-list* (NIL) mappings was introduced to ease incremental matching. During this process, model and image lines are aligned and, therefore, the input pose is refined. Similar to [19], this is performed by a weighted least squares technique minimizing

$$\sum_{k=1}^{m} e\left(M_{i_k}, I_{j_k}, (\boldsymbol{R}, \boldsymbol{t})\right)^2 \tag{23.3}$$

using an appropriate error function e. For example, if e measures the 3-D distance between a 3-D model line $M_{i_k} = \langle M_{i_k}^1, M_{i_k}^2 \rangle$ and the plane that is spanned by the corresponding image line $I_{j_k} = \langle I_{j_k}^1, I_{j_k}^2 \rangle$ and the center of projection (similarly to [20]), Eq. (23.3) transforms to

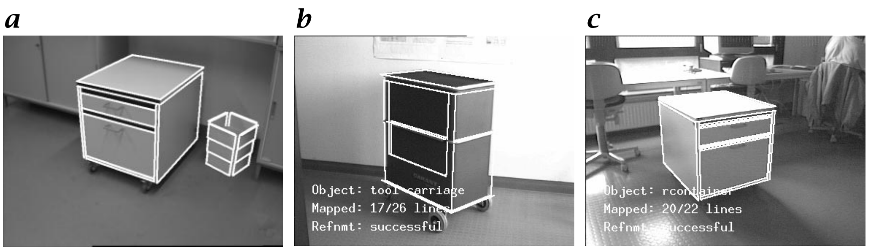
$$\sum_{k=1}^{m} W_{i_k} \sum_{l=1}^{2} \left[\tilde{\boldsymbol{n}}_{j_k}^T \boldsymbol{R}\left(M_{i_k}^l - \boldsymbol{t}\right)\right]^2$$

$$\tilde{\boldsymbol{n}}_{j_k} = \frac{\boldsymbol{n}_{j_k}}{||\boldsymbol{n}_{j_k}||}, \quad \boldsymbol{n}_{j_k} = \begin{bmatrix} I_{j_k}^1 \\ f \end{bmatrix} \times \begin{bmatrix} I_{j_k}^2 \\ f \end{bmatrix}$$

with f being the focal length of the CCD camera used.

If only coplanar features are visible, which are seen from a great distance compared to the size of the object, the 6 DOF estimation is quite unstable because some of the pose parameters are highly correlated. In this case *a priori* knowledge of the orientation of the camera with respect to the ground plane of the object might be used to determine two angular degrees of freedom. Naturally, this approach decreases the flexibility of the system. Tilted objects cannot be handled any longer. A more flexible solution is the use of a second image of the scene taken from a different viewpoint with known relative movement of the camera (*motion stereo*). By simultaneously aligning the model to both images the flat minimum of the 6 DOF estimation can be avoided. Note that for well-structured objects with some noncoplanar model features a 6 DOF estimation based on a single video image yields good results as well.

Some results for the object recognition are presented in Fig. 23.8. Details of the resulting pose (projected model superimposed in the original image) in Fig. 23.9 demonstrate the accuracy of the pose estima-



```
Object: tool carriage
Mapped: 17/26 lines
Refnmt: successful
```

```
Object: rcontainer
Mapped: 20/22 lines
Refnmt: successful
```

Figure 23.8: *Recognition of rigid 3-D objects with MORAL.*

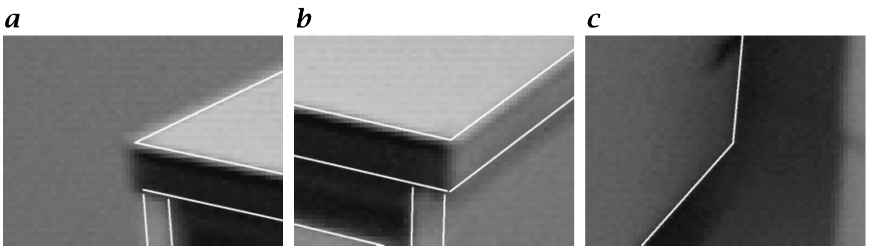


Figure 23.9: *Details of the computed pose in Fig. 23.8 a.*

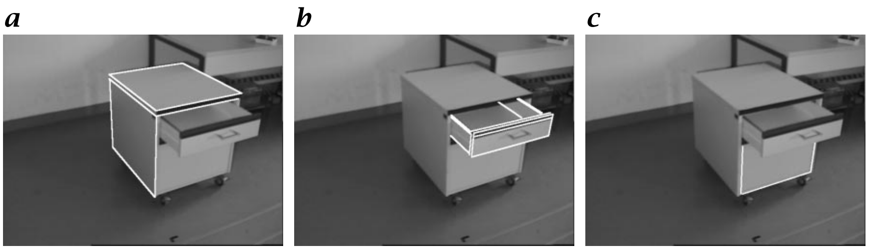


Figure 23.10: *Recognition of an articulated object: Guided by the hierarchical structure of the object model the unknown joint configuration of the drawer cabinet is determined recursively.*

tion. These tasks are performed in approximately 1 to 4 s on an Ultra SPARC-1/143, depending on the number of image and model features.

23.2.6 Articulated objects

Articulated objects are handled similarly to Hel-Or and Werman [21] using a hierarchical representation and recognition scheme [22]. The object model consists of rigid 3-D subcomponents linked by *joints*. Each joint state within the *joint configuration* of an object indicates the pose of a subcomponent relative to its parent component. The recognition task follows this tree-like structure by first estimating the 3-D pose of the static component (root) and afterwards determining the relative 3-D

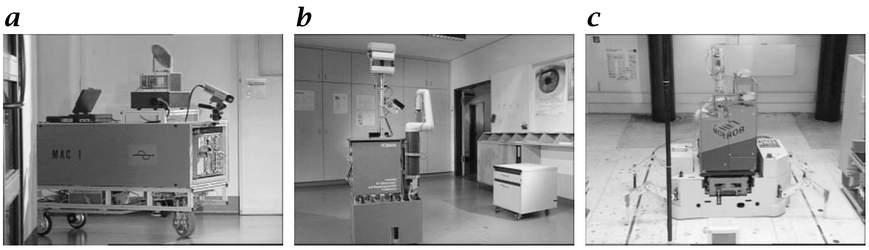


Figure 23.11: *Different AMS used for the experiments. Note that in the AMS shown in the camera is mounted at the robot hand.*

```
ok = moral_init(&ID)
ok = moral_load_param(ID,Configuration)
ok = moral_load_param(ID,CameraParameter)
ok = moral_load_object(ID,Object)
 ⋮
ok = moral_rec_object(ID,&Object,&Pose)
 ⋮
ok = moral_finish(ID)
```

Figure 23.12: *Typical RPC sequence.*

pose of the remaining components recursively. This method limits the search space for the actual correspondences between image and model features and copes with the problem of self-occlusion. In Fig. 23.10, the result of the determination of two translational joint states is shown: After recognizing the static component of the object DrawerCabinet, the translational joint states corresponding to the two drawers are computed following the object hierarchy.

23.3 Applications

In the following, several applications of MORAL in the context of autonomous mobile systems are described. The easy integration of MORAL into these different platforms (see Fig. 23.11) with different cameras and tasks shows the flexibility of the system. A typical sequence of RPC calls to MORAL is outlined in Fig. 23.12.

Two examples for the *vision-based navigation* of an AMS in a known laboratory environment with MORAL are shown in Fig. 23.13. Using the input pose (given by the odometry of the AMS), the predicted 3-D model features, and the image grabbed from the CCD camera, MORAL

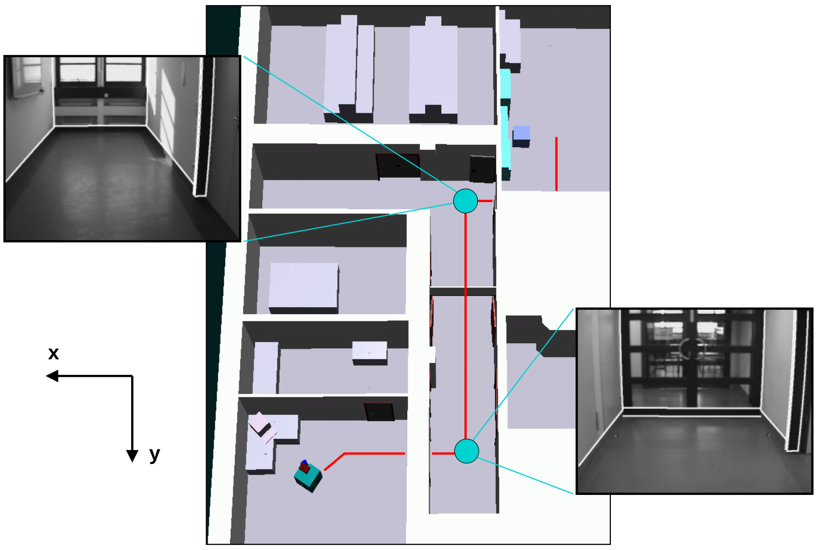


Figure 23.13: *Vision-based navigation in a laboratory environment.*

Table 23.1: *Discrepancy of the computed 3-D pose referring to a laser navigation unit.*

Error	Mean			Std.-Dev.		
	view dir.	perpend.	orient.	view dir.	perpend.	orient.
e_{2-D}	11.96 cm	3.75 cm	0.42°	6.39 cm	2.49 cm	0.31°
e_{3-D}	9.81 cm	6.27 cm	0.51°	6.39 cm	4.27 cm	0.41°

determines the pose of the AMS relative to the world. However, the uncertainty of the AMS pose leads to model-specific search spaces [18]. On an Ultra SPARC-1/143 workstation, the whole navigation task (image preprocessing and pose estimation) takes approximately 1 s. The precision of the computed absolute 3-D pose of the AMS during a drive through the corridor of a laboratory environment (see Fig. 23.13) is shown in Table 23.1. The values are measured referring to another localization unit based on a laser navigation sensor. Using two different error functions e_{2-D} and e_{3-D} (see Eq. (23.3)), the results show the quality of the computed 3-D pose. The positioning error is approximately 1 % of the average distance from the model features used. The result of the object recognition of a trashcan is shown in Fig. 23.14 a. Here, the rough object hypothesis was provided by the recognition module

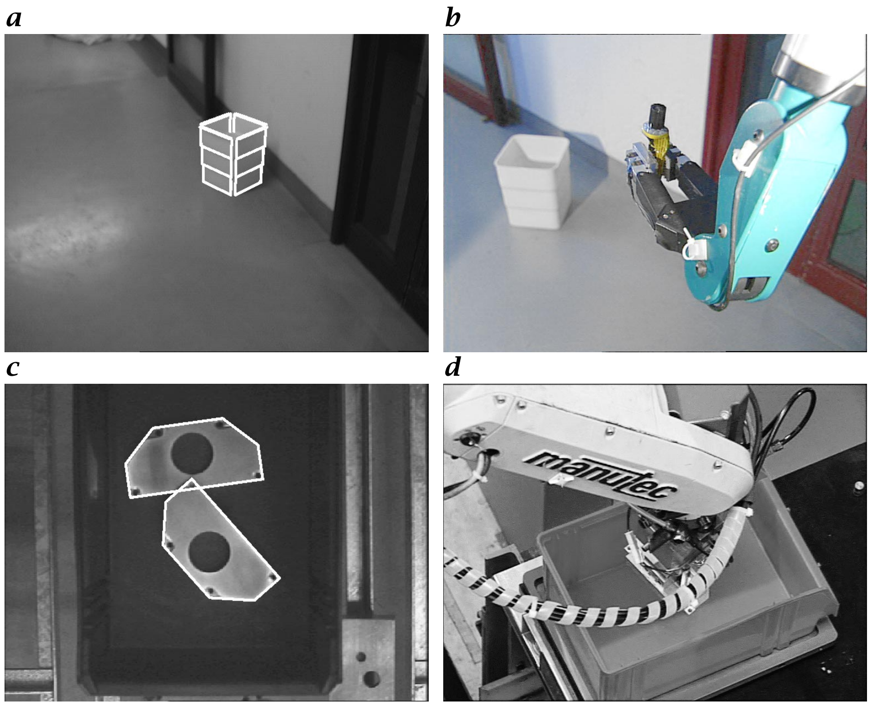


Figure 23.14: *Applications of MORAL in the context of autonomous mobile robots.*

(see Section 23.2.4). The computed 3-D pose of the trashcan enables the robot to dispose of some waste (Fig. 23.14 b).

A further application of MORAL—the grasping of a workpiece by a mobile robot—is presented in Fig. 23.14c,d. Because the exact pose of the workpiece is *a priori* unknown, the robot guiding system calls MORAL to determine the object pose. Using the CCD camera mounted in the gripper exchange system, MORAL computes the requested pose. In this case the *motion stereo* approach described in Section 23.2.5 is used to simultaneously align the model to two images [23].

Two applications of the recognition of articulated objects are shown in Fig. 23.15. In the upper row the result of the hierarchically recognition of the drawer cabinet (see Section 23.2.6) is shown enabling the robot to open the drawer. After recognizing the door frame the aim of the second task is to determine the opening angle (*rotational joint state*) of the door wing. Using this result, the robot can determine whether it can pass the door or has to open it.

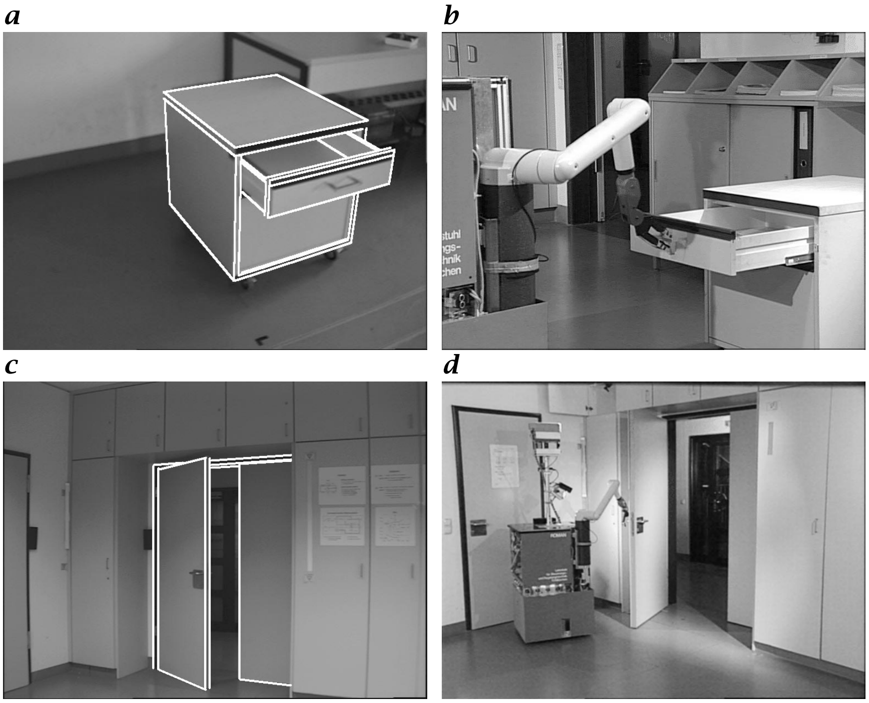


Figure 23.15: *Applications of the recognition of articulated 3-D objects.*

23.4 Conclusion

This chapter presented the object-recognition system MORAL. It is suitable for handling vision-based tasks in the context of autonomous mobile systems. Based on *a priori* known 3-D models, MORAL recognizes objects in single video images and determines their 3-D pose. Alternatively, the 3-D pose of the camera relative to the environment can also be computed. The flexibility of the approach has been demonstrated by several online experiments on different platforms.

Current work focuses on the following topics: using an indexing scheme, the object-recognition module should be able to handle a larger model database; in addition to the use of CAD models, the 3-D structure of objects should be reconstructed by images taken from different viewpoints; finally, curved lines are integrated as a new feature type.

Acknowledgment

The 3-D object recognition system MORAL presented in this chapter was developed within project L9, "Videobasierte Erfassung von Umweltinformationen durch ein autonomes, mobiles System" of the special research programme 331, “Informationsverarbeitung in autonomen, mobilen Handhabungssystemen” supported by the Deutsche Forschungsgemeinschaft (DFG).

23.5 References

[1] Lanser, S., Munkelt, O., and Zierl, C., (1995). Robust video-based object recognition using CAD models. In *Intelligent Autonomous Systems IAS-4, Karlsruhe, Germany, March 27-30*, U. Rembold, R. Dillmann, L. Hertzberger, and T. Kanade, eds., pp. 529–536. Amsterdam: IOS Press.

[2] Lanser, S., Zierl, C., Munkelt, O., and Radig, B., (1997). MORAL—A vision-based object recogntion system for autonomous mobile systems. In *CAIP, Kiel, Germany, September 10-12*, number 1296 in Lecture Notes in Computer Science, pp. 33–41. Berlin: Springer.

[3] Dickinson, S., Pentland, A., and Rosenfeld, A., (1992). From volumes to views: An approach to 3-D object recognition. *CVGIP: Image Understanding*, **55**(2):130–154.

[4] Grimson, W. E. L., (1990). Object recognition by constrained search. In *Machine Vision for Three-Dimensional Scenes*, H. Freeman, ed., pp. 73–108. Boston: Academic Press.

[5] Ikeuchi, K. and Kanade, T., (1988). Automatic generation of object recognition programs. *IEEE Trans. Computers*, **76**(8):1016–1035.

[6] Pope, A., (1994). *Model-based object recognition*. Technical Report TR-94-04, University of British Columbia.

[7] Fennema, C., Hanson, A., Riseman, E., Beveridge, J. R., and Kumar, R., (1990). Model-directed mobile robot navigation. *IEEE Trans. Systems, Man, and Cybernetics*, **20**(6):1352–1369.

[8] Christensen, H. and Kirkeby, N., (1994). Model-driven vision for indoor navigation. *Robotics and Autonomous Systems*, **12**:199–207.

[9] Kosaka, A. and Pan, J., (1995). Purdue experiments in model-based vision for hallway navigation. In *Workshop on Vision for Robots in IROS’95, Pittsburgh, PA, USA*, pp. 87–96. New York: IEEE Press.

[10] Lenz, R. and Tsai, R. Y., (1988). Techniques for calibration of the scale factor and image center for high accuracy 3-D machine vision metrology. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **10**(5):713–720.

[11] Lanser, S. and Zierl, C., (1995). Robuste Kalibrierung von CCD-Sensoren für autonome, mobile Systeme. In *Autonome Mobile Systeme, Karlsruhe, Germany, November 30 - December 1*, R. Dillmann, U. Rembold, and T. Lüth, eds., Informatik aktuell, pp. 172–181. Berlin: Springer.

[12] Wang, C. C., (1992). Extrinsic calibration of a vision sensor mounted on a robot. *Transactions on Robotics and Automation*, **8**(2):161–175.

[13] Hauck, A. and Stöffler, N., (1996). A hierarchical world model with sensor- and task-specific features. In *International Conference on Intelligent Robots and Systems (IROS)*, pp. 1614–1621. Osaka, Japan, Nov. 4-8.

[14] Eckstein, W. and Steger, C., (1997). Architecture for computer vision application development within the HORUS system. *Jour. Electronic Imaging*, **6**(2):244–261.

[15] Lanser, S. and Zierl, C., (1996). On the use of topological constraints within object recognition tasks. In *ICPR13, Vienna, Austria, August 25-30*, Vol. 1, pp. 580–584. Washington, DC: IEEE Computer Society Press.

[16] Lanser, S., (1997). *Modellbasierte Lokalisation gestützt auf monokulare Videobilder.* Dissertation, TU München, Fakultät für Informatik.

[17] Munkelt, O., (1995). Aspect-trees: generation and interpretation. *Computer Vision and Image Understanding*, **61**(3):365–386.

[18] Lanser, S. and Lengauer, T., (1995). On the selection of candidates for point and line correspondences. In *International Symposium on Computer Vision, Coral Gables, FL, USA, November 21-23*, pp. 157–162. Washington, DC: IEEE Computer Society Press.

[19] Lowe, D. G., (1991). Fitting parameterized three-dimensional models to images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **13**(5):441–450.

[20] Kumar, R. and Hanson, A. R., (1994). Robust methods for pose determination. In *NSF/ARPA Workshop on Performance versus Methodology in Computer Vision*, pp. 41–57. University of Washington, Seattle.

[21] Hel-Or, Y. and Werman, M., (1996). Constraint fusion for recognition and localization of articulated objects. *Int. J. Computer Vision*, **19**(1):5–28.

[22] Hauck, A., Lanser, S., and Zierl, C., (1997). Hierarchical recognition of articulated objects from single perspective views. In *CVPR, San Juan, Puerto Rico, June 17-19*, pp. 870–876. Washington, DC: IEEE Computer Society Press.

[23] Blessing, S., Lanser, S., and Zierl, C., (1996). Vision-based handling with a mobile robot. In *International Symposium on Robotics and Manufacturing (ISRAM)*, M. Jamshidi, F. Pin, and P. Dauchez, eds., Vol. 6, pp. 49–59. Montpellier, France, May 28-30: ASME Press, New York.

24 Flexible Models of Human Faces for the Analysis and Synthesis of Images

Thomas Vetter

Max-Planck-Institut für biologische Kybernetik, Tübingen, Germany

24.1 Introduction

When only a single image of a *face* is available, can we generate new images of the face across changes in viewpoint or illumination? The approach presented in this chapter acquires its knowledge about possible image changes from other faces and transfers this prior knowledge to a novel face image. Such learning strategies are well known by humans. In contrast, for *image synthesis* or image analysis little is known of how such a knowledge could be automatically acquired or how such a system could be implemented.

In recent years we have developed the concept of *linear object classes* and implemented an application for human faces [1, 2, 3]. The method allows us to compute novel views of a face from a single image. The method draws, on the one hand, on a general flexible face model that is learned automatically from examples and, on the other hand, on an algorithm that allows for matching this flexible model to a novel face image. In an *analysis by synthesis* loop the novel image is reconstructed

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

by the model. The novel image now can be described or coded through the internal model parameters that are necessary to reconstruct the image. The design of the model allows also for synthesizing new views of the face.

Analysis by synthesis. The requirement of pattern synthesis for pattern analysis has often been proposed within a Bayesian framework [4, 5] or has been formulated as an alignment technique [6]. This is in contrast to pure *bottom-up* techniques that have been advocated especially for the early stages of visual signal processing [7]. Here, a standard strategy is to reduce a signal to a feature vector and to compare this vector with those expected for signals in various categories. A crucial problem with these algorithms is that they cannot explicitly describe variations between or within the categories and, therefore, have difficulties separating unexpected noise from the variations within a particular category.

In contrast, the algorithm described in this chapter works by actively reconstructing the signal analyzed. Then, by comparing the real signal with its reconstruction it is decided whether or not the analysis is sufficient to explain the signal. Clearly, such an approach has the additional problem of defining a model function to reconstruct the input signal.

This chapter focuses on the analysis and synthesis of images of a specific object class, that is, on images of human faces. For object classes, such as faces or cars, where all objects share a common similarity, such a model function could be learned from examples. That is, the image model for the whole object class is derived by exploiting some prototypical example images.

Image models developed for the analysis and synthesis of images of a specific class of objects must solve two problems simultaneously: First, the model must be able to synthesize images that cover the whole range of possible images of the whole class; and second, it must be possible to match the model to a novel image.

In the past, 2-D image-based *flexible face model* have been constructed and applied for the synthesis of rigid and nonrigid face transformations [8, 9, 10]. These models exploit prior knowledge from example images of prototypical faces and work by building flexible image-based representations (*active shape models*) of known objects by a linear combination of labeled examples. The underlying coding of an image of a new object or face is based on linear combinations of the 2-D shape (warping fields) of examples of prototypical images as well as the linear combinations of their color values at corresponding locations (texture).

For the problem of synthesizing novel views to a single example image of a face, recently we developed the concept of *linear object classes*

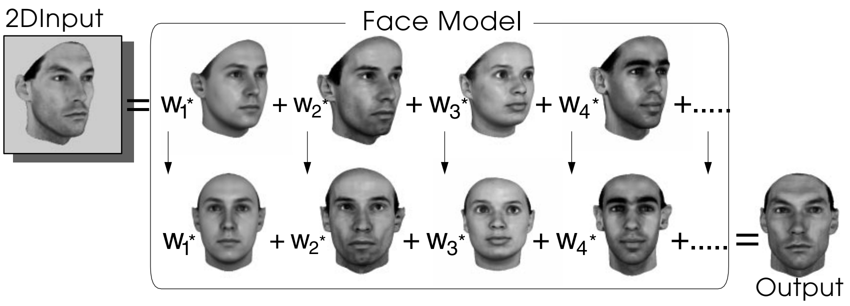

2DInput
Face Model
= W1* + W2* + W3* + W4* +.....
W1* + W2* + W3* + W4* +..... =
Output

Figure 24.1: *The basic concept for an example-based analysis and synthesis of face images. An input image of a face is analyzed by matching a face model to it, thus parameterizing the novel image in terms of a known face model. Depending on the expressive power of the face model, the parameters obtained can be used to predict new views of the face. Recently, face models have been constructed either by using several 2-D image models, each representing a specific viewing condition, or by directly using a flexible 3-D face model.*

[3]. This method allows us to compute novel views of a face from a single image. On the one hand, the method draws on a general flexible image model that can be learned automatically from example images, and, on the other hand, on an algorithm that allows matching this flexible model to a novel face image. The novel image now can be described or coded through the internal model parameters that are necessary to reconstruct the image. Such a coding allows also for synthesizing new views of the face, as shown in Fig. 24.1. Recently, we extended our method also to 3-D flexible face models [1].

For all these face models, it is crucial to establish the correspondence between each example face and a single reference face by matching image points in the 2-D approach and surface points in the 3-D case. Correspondence is a key step and it poses a difficult problem. However, for images of objects that share many common features, such as faces all seen from a single specific viewpoint, automated techniques seem feasible. Techniques applied in the past can be separated in two groups: one that establishes the correspondence for a small number of feature points only; and techniques computing the correspondence for every pixel in an image. For the first approach, usually models of particular features such as the eye corners or the whole chin line are developed off line and then matched to a new image [10, 11]. In this chapter we will focus on the second technique, which computes correspondence for each pixel in an image by comparing each image to a reference image [2, 12].

The chapter is organized as follows. First, we describe a flexible 3-D face model from which we will derive 2-D image models of faces.

Second, a method is described for computing dense correspondences between individual examples of human faces. Third, we describe an algorithm that allows matching the flexible face model to a novel image. Finally, we show examples of synthetic new images of a face computed from a single image.

24.2 Automated learning of flexible face models

Prior knowledge about faces can be captured in flexible face models that were developed in two as well as in three dimensions. Exploiting the general similarity among faces or face images, prototypical examples are linked to a general class model that captures regularities specific for this object class. The key to all flexible face models is that the set of example faces must be set in correspondence. Using a 3-D model for all vertices of the reference face, we have to find the corresponding vertex location on each face in the dataset. Similar for an image-based model for each pixel in the reference image, we have to find the corresponding pixel location on each face image in the dataset. In this section, first, the formal specification of flexible models is given and, second, an algorithm is described that allows for computing automatically the correspondence between the example faces. The proposed correspondence algorithm has two components: first, an optical flow algorithm that is used to approximate the pixel-by-pixel correspondence between all example faces; and second, the flexible face model itself. Known correspondence fields between prototypical faces are used to compute the correspondence to a novel face.

24.2.1 Formal specification of flexible face models

In the following we will start with a 3-D face model from which a 2-D model will be derived that was also developed independently in the image domain.

Three-dimensional models. In computer graphics, at present, the most realistic 3-D face representations consist of a 3-D mesh describing the geometry, and a texture map capturing the color data of a face. These representations of individual faces are obtained either by 3-D scanning devices or through photogrammetric techniques from several 2-D images of a specific face [13, 14]. Synthesis of new faces by interpolation between such face representation was already demonstrated in the pioneering work of Parke [14]. Recently, the idea of forming linear combinations of faces has been used and extended to a general 3-D flexible face model for the analysis and synthesis of 2-D facial images [1, 9, 15].

Shape model. The 3-D geometry of a face is represented by a shape-vector $\boldsymbol{S} = [X_1, Y_1, Z_1, X_2, \ldots, Y_n, Z_n]^T \in \mathbb{R}^{3n}$, that contains the X, Y, Z-coordinates of its n vertices. The central assumption for the formation of a flexible face model is that a set of M example faces $\boldsymbol{S}_i$ is available. Additionally, it is assumed that all these example faces $\boldsymbol{S}_i$ consist of the same number of n consistently labeled vertices, in other words, all example faces are in full correspondence (see next section on correspondence). Usually, this labeling is defined on an average face shape, which is obtained iteratively, and which is often denoted as reference face $\boldsymbol{S}_{\text{ref}}$. Additionally, all faces are assumed to be aligned in an optimal way by rotating and translating them in 3-D space. Under this assumptions a new face geometry $\boldsymbol{S}_{\text{model}}$ can be generated as a linear combination of M example shape-vectors $\boldsymbol{S}_i$ each weighted by c_i

$$\boldsymbol{S}_{\text{model}} = \sum_{i=1}^{M} c_i\, \boldsymbol{S}_i \tag{24.1}$$

The linear shape model allows for approximating any given shape $\boldsymbol{S}$ as a linear combination with coefficients that are computed by projecting $\boldsymbol{S}$ on the example shapes $\boldsymbol{S}_i$. The coefficients c_i of the projection then define a coding of the original shape vector in this vector space that is spanned by all examples.

Texture model. The second component of a flexible 3-D face or head model is texture information, which is usually stored in a *texture map*. A texture map is simply a 2-D color pattern storing the brightness or color values (ideally only the *albedo* of the surface). A u, v coordinate system associates the texture map with the modeled surface. The texture map is defined in the 2-D u, v coordinate system. For polygonal surfaces as defined through a shape vector $\boldsymbol{S}$, each vertex has an assigned u, v texture coordinate. Between vertices, u, v coordinates are interpolated. For convenience we assume that the total number n of stored values in such a texture map, is equal to the total number of vertices in a shape vector $\mathbf{S}$.

The linear texture model, described by Choi et al. [9] starts from a set of M example face textures $\boldsymbol{T}_i$. Similar to the shape model described earlier, it is assumed that all M textures $\boldsymbol{T}_i$ consist of the same number of n consistently labeled texture values, that is, all textures are in full correspondence. For texture synthesis linear models are used again. A new texture $\boldsymbol{T}_{\text{model}}$ is generated as the weighted sum of M given example textures $\boldsymbol{T}_i$ as follows:

$$\boldsymbol{T}_{\text{model}} = \sum_{i=1}^{M} b_i \boldsymbol{T}_i \tag{24.2}$$

Similar to the linear expansion of shape vectors (Eq. (24.1)), the linear expansion of textures can be understood and used as an efficient coding schema for textures. A new texture can be coded by its M projection coefficients b_i in the "texture vector space" spanned by M basis textures.

Two-dimensional models. A flexible 2-D face model formally can be derived by projecting the flexible 3-D face model (Eqs. (24.1) and (24.2)) into the image domain for a fixed projection and illumination condition [1, 3]. For the 3-D example faces we obtain the face images $\boldsymbol{G}_0, \boldsymbol{G}_1, \ldots, \boldsymbol{G}_M$. Let $\boldsymbol{G}_0$ be the reference image, and let positions within $\boldsymbol{G}_0$ be parameterized by (u, v).

Pixelwise correspondences between $\boldsymbol{G}_0$ and each example image are mappings $\boldsymbol{s}_j : \mathbb{R}^2 \to \mathbb{R}^2$, which map the points of $\boldsymbol{G}_0$ onto $\boldsymbol{G}_j$, that is, $\boldsymbol{s}_j(u, v) = (x, y)$, where (x, y) is the point in $\boldsymbol{G}_j$ that corresponds to (u, v) in $\boldsymbol{G}_0$. An overview on techniques for image warping is given in Volume 2, Chapter 9. We refer to $\boldsymbol{s}_j$ as a *correspondence field* and interchangeably as the *2-D shape vector* for the vectorized $\boldsymbol{G}_j$. The 2-D correspondence s_j between each pixel in the rendered reference image $\boldsymbol{G}_0$ and its corresponding location in each rendered example image $\boldsymbol{G}_i$, can be directly computed as the projection P of the differences of 3-D shapes between all 3-D faces and the reference face. *Warping* image $\boldsymbol{G}_j$ onto the reference image $\boldsymbol{G}_0$, we obtain $\boldsymbol{t}_j$ as

$$\boldsymbol{t}_j(u, v) = \boldsymbol{G}_j \circ \boldsymbol{s}_j(u, v) \Leftrightarrow \boldsymbol{G}_j(x, y) = \boldsymbol{t}_j \circ \boldsymbol{s}_j^{-1}(x, y)$$

Thus, $\{\boldsymbol{t}_j\}$ is the set of shape-normalized prototype images, referred to as *texture vectors*. They are normalized in the sense that their shape is the same as the shape of the chosen reference image.
The flexible image model is the set of images I^{model} parameterized by $\boldsymbol{c} = [c_0, c_1, \ldots, c_M]^T$, $\boldsymbol{b} = [b_0, b_1, \ldots, b_M]^T$ such that

$$I^{\text{model}} \circ \left(\sum_{i=0}^{M} c_i \boldsymbol{s}_i\right) = \sum_{j=0}^{M} b_j \boldsymbol{t}_j \tag{24.3}$$

The summation $\sum_{i=0}^{M} c_i \boldsymbol{s}_i$ constrains the 2-D shape of every model image to be a linear combination of the example 2-D shapes. Similarly, the summation $\sum_{j=0}^{M} b_j \boldsymbol{t}_j$ constrains the texture of every model image to be a linear combination of the example textures.

For any values for c_i and b_i, a model image can be rendered by computing $(x, y) = \sum_{i=0}^{M} c_i \boldsymbol{s}_i(u, v)$ and $g = \sum_{j=0}^{M} b_j \boldsymbol{t}_j(u, v)$ for each (u, v) in the reference image. Then the (x, y) pixel is rendered by assigning $I^{\text{model}}(x, y) = g$, that is, by warping the texture into the model shape.

24.2.2 Matching the flexible face model to an image

A novel face image can be analyzed by matching a flexible face model and thus parameterizing the novel image in terms of the known model

[2, 16]. Matching the image model as defined in Eq. (24.3) to a novel image leads to an error function of the model parameters $\boldsymbol{c}$ and $\boldsymbol{b}$

$$E(\boldsymbol{c}, \boldsymbol{b}) = \frac{1}{2} \sum_{x,y} [I^{\text{novel}}(x, y) - I^{\text{model}}(x, y)]^2$$

In order to compute I^{model} (see Eq. (24.3)) the shape transformation $(\sum c_i \boldsymbol{s}_i)$ has to be inverted or one has to work in the coordinate system (u, v) of the reference image, which is computationally more efficient. Therefore, the shape transformation (given some estimated values for $\boldsymbol{c}$ and $\boldsymbol{b}$) is applied to both I^{novel} and I^{model}. From Eq. (24.3) we obtain

$$E = \frac{1}{2} \sum_{u,v} [I^{\text{novel}} \circ (\sum_{i=0}^{M} c_i \boldsymbol{s}_i(u, v)) - \sum_{j=0}^{M} b_j \boldsymbol{t}_j(u, v)]^2$$

Minimizing the error yields the model image that best fits the novel image with respect to the L_2 norm. The optimal model parameters $\boldsymbol{c}$ and $\boldsymbol{b}$ are found by a stochastic gradient descent algorithm [17]. The robustness of the algorithm is improved using a coarse-to-fine approach [18]. In addition to the textural pyramids, separate resolution pyramids are computed for displacement fields $\boldsymbol{s}$ in x and y.

24.2.3 Correspondence by optical flow algorithms

Dense, pixel-by-pixel correspondence fields between face images were computed for the first time by Beymer et al. [8] using an optical flow algorithm. Unlike temporal sequences taken from one scene, a comparison of images of completely different scenes or faces may violate a number of important assumptions made in optical flow estimation. However, some optical flow algorithms can still cope with this more difficult matching problem. For more details on correspondence and optical flow see Volume 2, Chapter 13.

In previous studies [3], we built flexible image models of faces based on correspondence between images, using a coarse-to-fine gradient-based method [19] and following an implementation described in [20]. The optimization problem given in Eq. (13.13), Volume 2, Section 13.3.1 is solved for a 5×5 pixel neighborhood in a single iteration. However, as this is only a crude approximation to the overall matching problem, an iterative coarse-to-fine strategy is required. The algorithm starts with an estimation of correspondence on low-resolution versions of the two input images. The resulting flow field is used as an initial value to the computation on the next higher level of resolution. Iteratively, the algorithm proceeds to full resolution. In our applications, results were dramatically improved if images on each level of resolution were computed not only by downsampling the original (Gaussian pyramid),

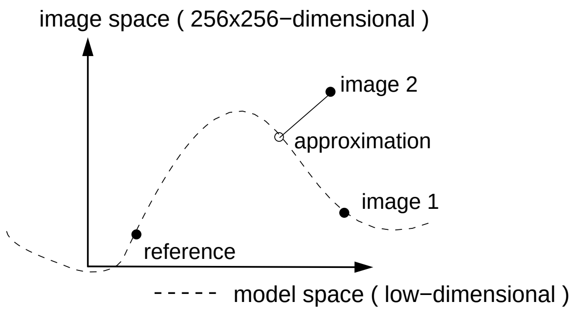


image space (256x256−dimensional)
image 2
approximation
image 1
reference
----- model space (low−dimensional)

Figure 24.2: *Given the flexible model provided by the combination of a reference image and image 1 (in correspondence), the goal is to find the correspondence between the reference and image 2. The solution is to find first the linear combination of the reference and image 1 that is the image closest to image 2 (its approximation). Then find the correspondences from this approximation to image 2 using optical flow. The two flow fields can then be composed to yield the desired flow from the reference image to image 2.*

but also by bandpass filtering (Laplacian pyramid). The Laplacian pyramid was computed from the Gaussian pyramid adopting the algorithm proposed by Burt and Adelson [18], multiresolution representations are described in detail in Volume II, Chapter 10.

The adaptation and extension of this technique to 3-D face data is straightforward due to the fact that these 2-D manifolds can be parameterized in terms of two variables. For example, in a cylindrical representation, faces are described by radius and color values at each angle ϕ and height h. Images, on the other hand, consist of gray-level values in image coordinates x,y. Thus, in both cases correspondence can be expressed by a mapping $C : \mathbb{R}^2 \to \mathbb{R}^2$ in parameter space (for more details, see Vetter and Blanz [1]).

24.2.4 Automated model learning

Here, a bootstrapping algorithm is described that is capable of computing correspondence between a prototypical set of faces without the need of human assistance [2]. The basic idea is to combine the matching of the face model with the optical flow algorithm for the correspondence computation. Suppose that an existing flexible model is not powerful enough to match a new image and thereby find correspondence with it. The idea is first to find rough correspondences to the novel image using the (inadequate) flexible model and then to improve these correspondences by using an optical flow algorithm. This idea is illustrated in Fig. 24.2. Suppose a model consists only of a reference image and image 1 (and the pixelwise correspondences between them).

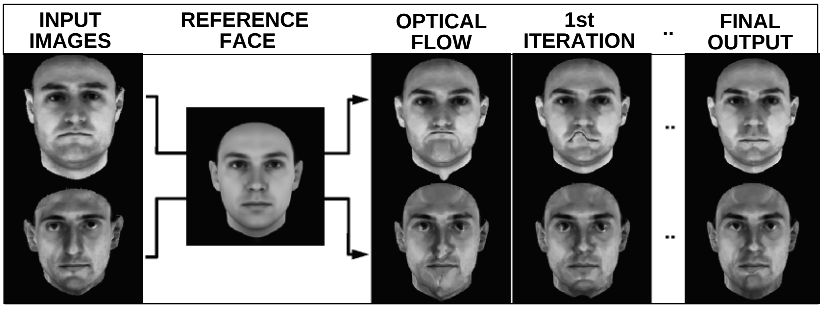


Figure 24.3: *Bootstrapping the correspondence. The correspondence between face images (left column) and a reference face can be visualized by backward warping of the face images onto the reference image (three columns on the right). This should produce images that should have the geometry of the reference face and the pixel values of the input. The correspondence obtained through the optical flow algorithm does not allow a correct mapping (center column). The first iteration with a linear flexible model consisting of two principal components already yields a significant improvement (top row). After four iterations with 10, 30 and 80 components, respectively, all correspondences were correct (right column).*

The correspondence between this model and image 2 can be computed by matching the model to image 2 obtaining an approximation of image 2. The correspondences are then improved by running an optical flow algorithm between the intermediate approximation image and image 2. The final correspondence between the reference image and image 2 can be computed by combining the correspondence between the reference and the approximation and that between the approximation and image 2.

In applications on human faces as well as on handwritten digits the following implementation was used (for more details, see Vetter et al. [2]) to build flexible models. Starting on a set of N images without correspondence, a reference image was computed first. The correspondence between an arbitrary example image and all other images of the set was computed by the optical flow algorithm. Combining the average of all correspondence fields $\boldsymbol{s}_i$ with the average of all textures $\boldsymbol{t}_i$ results in a first approximation of the average image of the set. By iterating this procedure we obtained a stable average. However, the correspondences computed by the optical flow between this average and the examples were only correct in 80% of the cases. In a second step these correspondences were improved by our "bootstrapping algorithm." First, a small flexible image model was built on the basis of the first two statistically most significant modes of a principal component analysis (PCA) performed separately on the correspondence fields

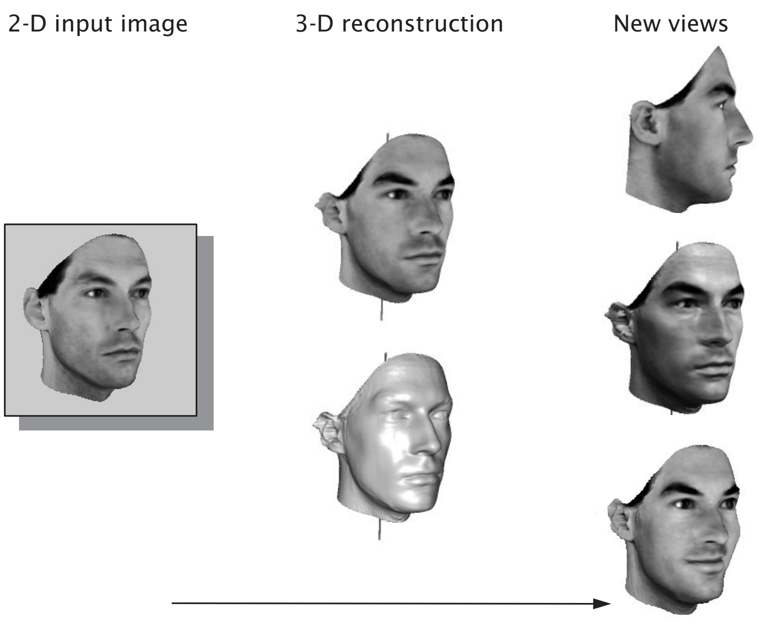


Figure 24.4: *Three-dimensional reconstruction of a face (center) from a single 2-D image of known orientation and illumination (left). The prior knowledge about faces was given through a training set of 3-D data of 100 faces different from the input face. The right column shows new views generated by the flexible face model. From the top to bottom, the face is shown from a new viewpoint, under a changed illumination and smiling (modified from [1]).*

$\boldsymbol{s}_i$ and the textures $\boldsymbol{t}_i$. Second, after approximating each image with this model the correspondence was improved using the optical flow algorithm. This PCA procedure and the refinement of correspondences was repeated several times simultaneously increasing the number of principal components used. The results were stable and correct correspondence fields. Examples of such an iteration are shown in Fig. 24.3.

24.3 View synthesis

Novel views from a single 2-D face image can be generated by matching a flexible face model to the image. Synthetically rotated face images were generated based on image-based flexible face models [3] as well as using a 3-D face model [1]. Figure 24.4 shows an example of a 3-D face reconstruction from a single image of known orientation and illumination. Matching the 2-D image model to the novel image, the 2-D model parameters $\boldsymbol{c}$ and $\boldsymbol{b}$ can be used in the 3-D flexible face model as defined in Eqs. (24.1) and (24.2) (for more details, see Vetter and

Blanz [1]). The output of the 3-D flexible face model is an estimate of the 3-D shape from the 2-D image. This 3-D face reconstruction allows us to render new images from any viewpoint or under any illumination condition. Additionally, nonrigid face transitions can be mapped onto the 3-D face model. The transformation for a smile is extracted from another person and then mapped onto the face of a different person. Computing the correspondence between two examples of one person's face, one example showing the face smiling and the other showing the face in a neutral expression results in a correspondence field or deformation field that captures the spatial displacement for each vertex in the model according to the smile. Such a "smile-vector" now can be added or subtracted from each face that is in correspondence to one of the originals, making a neutral-looking face more smiling or giving a smiling face a more emotionless expression.

Data set. All image synthesis experiments we performed on images rendered from a 3-D data set obtained from 200 laser scanned (Cyberware^{TM}) heads of young adults (100 male and 100 female). The laser scans provide head structure data in a cylindrical representation, with radii of surface points sampled at 512 equally spaced angles, and at 512 equally spaced vertical distances. Additionally, the RGB-color values were recorded in the same spatial resolution and were stored in a texture map with 8 bits per channel. All faces were without makeup, accessories, and facial hair. After the head hair was removed digitally, individual heads were represented by approximately 70,000 vertices and the same number of color values.

The data set of 200 faces was split randomly into a training and a test set, each consisting of 100 faces. The training set was used to "learn" a flexible model. From the test set, images were rendered showing the faces 30° from frontal, and using mainly ambient light. The image size used in the experiments was 256-by-256 pixels and 8 bits per color channel.

Results. Evaluating the 3-D face reconstructions from 100 reconstructions, 72 faces were highly similar and often hard to distinguish from the original in a 30° view. In 27 cases, persons were still easy to identify, but images displayed some differences, for example, in the shape of cheeks or jaw. Only one reconstruction was showing a face clearly unlike the original, yet a face that could very well exist.

We rated the example shown in Fig. 24.4 as highly similar, but within this category it is average. While the reconstructed head appears very similar to the original image in a 30° view, it is not surprising to notice that front and profile views reveal a number of differences.

24.4 Conclusions

Flexible models as described in this paper can be understood as a special case of deformable templates [21]. Deformable templates are specified by a set of parameters that enable *a priori* knowledge about the expected shape of a feature to guide the matching process. The chosen parameterization of the templates has been shown to be the most critical step, often hindering the discovery of an optimal solution.

The improvement of the methods described is that the parameterization is learned directly from a given example set of images or objects of specific class. The key is a correspondence-based representation in which images of an object class are modeled in two separate linear vector spaces, one containing the shape information and the other texture information. The vector space property allows us to solve the two main problems in an analysis by synthesis approach. Exploiting the statistics of the example images, a parameterization can be derived that explicitly reflects the image variations of the class. On the other hand, these explicit parameters allow a continuous modeling of images, the requirement necessary for matching a model to a novel image.

The key requirement of the approach is the correspondence between the set of example faces. The bootstrapping approach for the automated model formation as described can not be a full answer to the problem of computing correspondence between prototypes. It does, however, provide an initial and promising solution to the very difficult problem of an automatic synthesis of flexible models from a set of prototype examples.

We presented a method for approximating the 3-D shape of a face from just a single image. In an analysis-by-synthesis loop a flexible 3-D-face model is matched to a novel image. The novel image now can be described or coded through the model parameters reconstructing the image. Prior knowledge on the 3-D appearance of faces derived from an example set of faces allows predicting new images of a face.

Clearly, conditions in our current matching experiments were simplified in two ways. First, all images were rendered from our test set of 3-D-face scans. Second, projection parameters and illumination conditions were known. The extension of the method to face images taken under arbitrary conditions, in particular, to any photograph, will need several improvements. On the one hand, adding more free parameters into the matching procedure will require more sophisticated model representations, especially in terms of the statistical dependence of the parameters. On the other hand, the linear model depends on the given example set. In order to represent faces from a different race or a different age group, the model will need examples of these, an effect also well known in human perception (see, e.g., [22]).

While the construction of 3-D-models from a single image is very difficult and often an ill-posed problem in a bottom-up approach, our example-based technique allows us to obtain satisfying results by means of a maximum likelihood estimate. The ambiguity of the problem is reduced when several images of an object are available, a fact that is exploited in stereo or motion-based techniques. In our framework, we can make use of this additional information by simultaneously optimizing the model parameters for all images, while the camera and lighting parameters are adjusted for each image separately. The method presented in this paper appears to be complementary to nonmodel-based techniques such as stereo. While our approach is limited to results within a fixed model space, these techniques are often not reliable in areas with little structure. For the future, we plan to combine both techniques to avoid the disadvantages of each.

24.5 References

[1] Vetter, T. and Blanz, V., (1998). Estimating coloured 3-D face models from single images: An example based approach. In *Computer Vision—ECCV'98*. Freiburg, Germany: Lecture Notes in Computer Science. New York: Springer.

[2] Vetter, T., Jones, M. J., and Poggio, T., (1997). A bootstrapping algorithm for learning linear models of object classes. In *IEEE Conference on Computer Vision and Pattern Recognition—CVPR'97*. Puerto Rico, USA: Los Alamitos, CA: IEEE Computer Society Press.

[3] Vetter, T. and Poggio, T., (1997). Linear object classes and image synthesis from a single example image. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **19**(7):733–742.

[4] Grenander, U., (1978). *Pattern Analysis, Lectures in Pattern Theory*, first edition. New York: Springer.

[5] Mumford, D., (1996). Pattern theory: a unifying perspective. In *Perception as Bayesian Inference*, D. Knill and W. Richards, eds. Cambridge: Cambridge University Press.

[6] Ullman, S., (1989). Aligning pictorial descriptions: An approach for object recognition. *Cognition*, **32**:193–254.

[7] Marr, D., (1982). *Vision*. San Fancisco: W. H. Freeman.

[8] Beymer, D., Shashua, A., and Poggio, T., (1993). *Example-based image analysis and synthesis*. A.I. Memo No. 1431, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

[9] Choi, C., Okazaki, T., Harashima, H., and Takebe, T., (1991). A system of analyzing and synthesizing facial images. In *Proc. IEEE Int. Symposium of Circuit and Systems (ISCAS91)*, pp. 2665–2668.

[10] Lanitis, A., Taylor, C., Cootes, T., and Ahmad, T., (1995). Automatic interpretation of human faces and hand gestures using flexible models. In

Proc. International Workshop on Face and Gesture Recognition, M. Bichsel, ed., pp. 98–103. Zürich, Switzerland.

[11] Herpers, R., Michaelis, M., Lichtenauer, K. H., and Sommer, G., (1996). Edge and keypoint detection in facial regions. In *Proc. International Conference on Automatic Face and Gesture Recognition*, pp. 22–27. Killington, VT.

[12] Beymer, D. and Poggio, T., (1996). Image representation for visual learning. *Science*, **272**:1905–1909.

[13] Akimoto, T., Suenaga, Y., and Wallace, R., (1993). Automatic creation of 3-D facial models. *IEEE Computer Graphics and Applications*, **13**(3):16–22.

[14] Parke, F., (1974). *A Parametric Model of Human Faces*. Doctoral thesis, University of Utah, Salt Lake City. UT.

[15] Poggio, T. and Vetter, T., (1992). *Recognition and structure from one 2-D model view: observations on prototypes, object classes, and symmetries*. A.I. Memo No. 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

[16] Jones, M. and Poggio, T., (1996). *Model-based matching by linear combination of prototypes*. A.I. Memo No. 1583, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

[17] Viola, P., (1995). *Alignment by maximization of mutual information*. A.I. Memo No. 1548, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

[18] Burt, P. and Adelson, E., (1983). The Laplacian pyramide as a compact image code. *IEEE Trans. Communications*, **31**:532–540.

[19] Bergen, J., Anandan, P., Hanna, K., and Hingorani, R., (1992). Hierarchical model-based motion estimation. In *Proceedings of the European Conference on Computer Vision*, pp. 237–252. Santa Margherita Ligure, Italy.

[20] Bergen, J. and Hingorani, R., (1990). *Hierarchical motion-based frame rate conversion*. Technical report, David Sarnoff Research Center Princeton NJ 08540.

[21] Yuille, A. L., Hallinan, P. W., and Cohen, D. S., (1992). Feature extraction from faces using deformable templates. *International Jour. Computer Vision*, **8**(2):99–111.

[22] O'Toole, A., Deffenbacher, K., Valentin, D., and Abdi, H., (1994). Structural aspects of face recognition and the other-race effect. *Memory and Cognition*, **22**:208–224.

25 Knowledge-Based Image Retrieval

Thorsten Hermes[1], Christoph Klauck[2], and Otthein Herzog[1]

[1]Technologie-Zentrum Informatik, Universität Bremen
[2]Fachbereich Elektrotechnik und Informatik, Fachhochschule Hamburg

25.1 Introduction

In order to retrieve a set of intended images from an image archive, human beings think of special contents with respect to the searched scene, such as a countryside picture or a technical drawing. The necessity of a semantics-based retrieval language leads to a content-based analysis and retrieval of images. From this point of view, our project Image Retrieval for Information Systems (IRIS) develops and combines methods and techniques of computer vision and *knowledge representation* in a new way in order to automatically generate textual *content descriptions* of images.

First, in this chapter we provide a rough overview of existing image retrieval systems (Section 25.1.1) and, additionally, as we introduced in Alshuth et al. [1, 2] how *video retrieval* can be treated as still image retrieval, we provide an overview of existing video retrieval systems (Section 25.1.2).

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

Later, we introduce the architecture of the system IRIS (Section 25.2), and present the methods of *feature extraction* with respect to color (Section 25.2.1), texture (Section 25.2.2), and contour segmentation (Section 25.2.3). We concentrate on the discussion of formalization knowledge for modeling concepts and *object recognition* (Section 25.3). Our approach is very similar to that by Niemann (Chapter 27). The differences are related to the *inference machine*. While Niemann uses the inference mechanism of semantic networks, we compile the knowledge of a special semantic network into a *graph grammar* (rules) and use in this sense a *rule-based inference mechanism*. Finally, we give examples for the separate analysis task applied to a landscape scene (Section 25.4). The system is implemented on IBM RS/6000 using AIX and on Windows NT. It has been tested with an archive comprising 1200 pictures as well as with technical drawings, computer-aided design (CAD). An IRIS-System demo version can be found on the corresponding CD-ROM to this handbook.

25.1.1 Image retrieval systems

Content-based image retrieval has been a very popular research topic for the last few years. With the ongoing development of multimedia technology, the number of information systems containing image retrieval functions is increasing rapidly. Surveys and discussions of approaches and systems for image retrieval have been published by Nagy [3], Cawkell [4], and Jain [5].

The whole image retrieval process can be divided into two stages: generating the *image annotations* and retrieving images. An early idea for image annotation is the text description generated by users, for example, the title and caption of the images as well as some additional description done by the users [6, 7, 8]. This approach is restricted by the effort of manual annotation and the user-dependent differences in the annotations. This leads very soon to inconsistencies in annotations and to an undesired requirement: the person formulating the queries has to acquire knowledge concerning the criteria of the annotation generation. Furthermore, it is a considerable and expensive effort to manually generate the content descriptions for thousands of images.

Therefore, there is an increasing interest in developing feature-based image retrieval systems that integrate technologies of computer vision in extracting image features into the process of the annotation generation. There are two kinds of research groups carrying out content-based image retrieval. The first one uses the well-known computer vision algorithms for generating object description and stores them in a textual form. These annotations are then transferred to a retrieval system. The system IRIS developed by Hermes et al. [9] belongs to this group.

The second approach emphasizes interactively matching similar objects based on shape, color, and texture [10, 11, 12, 13, 14, 15]. Picard and Minka [16] use texture as a feature in their content-based retrieval method. First, they define semantic-based labels for different regions of an image and then compute the parameters of the texture models within that region. Rather than using only one texture model, they employ several models and select the best one for discriminating a region from the rest of the image.

Another example of image retrieval by matching image examples or user-drawn shapes is the QBIC project [13]. The image features used are color, texture, and shape, together with some geometrical parameters of regions. The image features that provide the query information are semiautomatically extracted.

Until now, a major breakthrough in image retrieval has not been achieved. The following are key ideas that need to be addressed successfully for content-based retrieval systems to become a reality:

- definition of relevant image features and automated analysis of images;
- application of knowledge to special domains; and
- development of powerful user interfaces for interactive querying.

There is no single universal approach to the content-based retrieval problems, even in a restricted application domain. Therefore, we need *domain knowledge* for special application. Chakravarthy [8] uses semantics-based relations in information retrieval. Semantics-based relations provide a way of expressing knowledge about the world. Several examples are presented to illustrate the proper incorporation of semantics-based relations into the process of matching queries to the picture representations.

This chapter describes the IRIS system, an image retrieval system combining automatic analysis methods of computer vision with *knowledge-based region matching*. The image analysis approach taken in this project combines color, texture, and contour as a base for the *object recognition*. The resulting *content descriptions* are stored in a textual form for automatic retrieval that can be realized by any text retrieval system. Furthermore, the domain knowledge can be developed and maintained separately from the main system in a knowledge base. In this way, we give the users the opportunity to build and select the suitable knowledge base for their application.

25.1.2 Video retrieval systems

Present-day image archives are built up using only manual annotation, that is, applying pre-defined key words from a homogeneous input of categories and brief descriptions. In this way, however, details within the image will be ignored. As compared to image archives, video archives offer a better opportunity for queries, because, besides mere information about images, queries concerning camera angles, sequences of scenes, etc., may be formulated. There are two different approaches using a fixed pattern and, thus, trying to classify the image information in order to gain time and facilitate the search:

- Every tenth *frame* of a video will be described. Thus, access to image information is possible without previous classification, and there is no additional information necessary with respect to a video clip. However, a content-based classification is not possible.
- Using color histograms, changes of scenes will be located and analyzed [17], applying the *DCT coefficient* for videos coded in MPEG [18, 19]. Access to image information of shots can be gained with the two methods below:
  - the rough contents of an image sequence can be derived from *key frames*. A key frame is a single characteristic frame of a sequence. This image is the basis for the annotation; and
  - with a dynamical mosaicing technique, a single total image may be generated from a sequence [20], and can be analyzed for a subsequent query. However, the dynamic video semantics cannot be captured in this way.

Some approaches of existing *multimedia* systems are based on the retrieval of image information from libraries (e.g., maps), but only few have been developed for tracing video data.

Query by example. Applying image processing methods, the queries in this first group are carried out by examples, rough sketches of an object, or search for components (contours, textures, shape, or layout). Examples:

PHOTOBOOK. This system consists of a selection of interactive tools for viewing and searching images and image sequences. The PHOTOBOOK approach is characterized by a broad selection of possible input data, such as gray-scale images, color images, images containing only edges or textures, and, in addition, voxel images.

ART MUSEUM is based on the concept that paintings and pictures may be classified according to stylistic elements [21].

QBIC. The QBIC system offers a search on the basis of individual images and sequences considering one data source only (individual images or videos) at a time [13].

Iconic query. The visual user-system-interface of the second group is considerably more distinct: with a manual input the icons represent different types of images that are the basis for a classification. As these are subdivided into classes, the user may direct his/her inputs and queries to the system.

VIRTUAL VIDEO BROWSER is a movie browser based on icons representing the different categories of movies and video clips [22].

VIDEO DATABASE BROWSER (VDB) allows a user to select individual videos and sequences with a mixed query interface [23].

MEDIA STREAMS is based on a visual icons language and an interlocking representation of video data [24].

Because of the necessary extensive access to the individual image information during the search process, the forementioned systems require considerable computational resources to avoid unacceptable response times.

Additional systems. Special areas of retrieval in distributed multimedia libraries are covered by the following additional systems:

OMNIS. The OMNIS Digital Library System that has been developed at TU Munich is a system for the management of text documents within a system of distributed libraries [25]. Both official documents (textbooks, proceedings) and "gray" literature, for example, internal reports and prepublications, are being registered.

With the semiautomatic video database system of the University PENNSYLVANIA, video sequences can be detected automatically with the aid of shot recognition methods [26].

ELINOR. The ELINOR Electronic-Library-System of the De-Montfort University administers teaching material of the university with a special view to the requirements of the student-user group [27]. The annotated material consists of textbooks, periodicals, scripts, films, etc.

VideoSTAR. The VideoSTAR-System at the Norwegian Institute of Technology stores video documents in a database detecting automatically the individual shots by standard procedures (histogram evaluation) [28].

As shown in the foregoing, neither the currently available video retrieval and annotation systems nor those being developed display an automatic annotation.

Up to now only the automatic recognition of individual scenes has been possible, and, here, the analysis is restricted to special features, for example, colors (Univ. of Pennsylvania).

As a rule, the user carries out the content-based description of images and video sequences, respectively, (VideoSTAR, ELINOR). Thus current systems are very time-consuming for the user and at the same time demand profound background knowledge. Furthermore, with two users annotating a similar image, the risk of faulty annotations in-

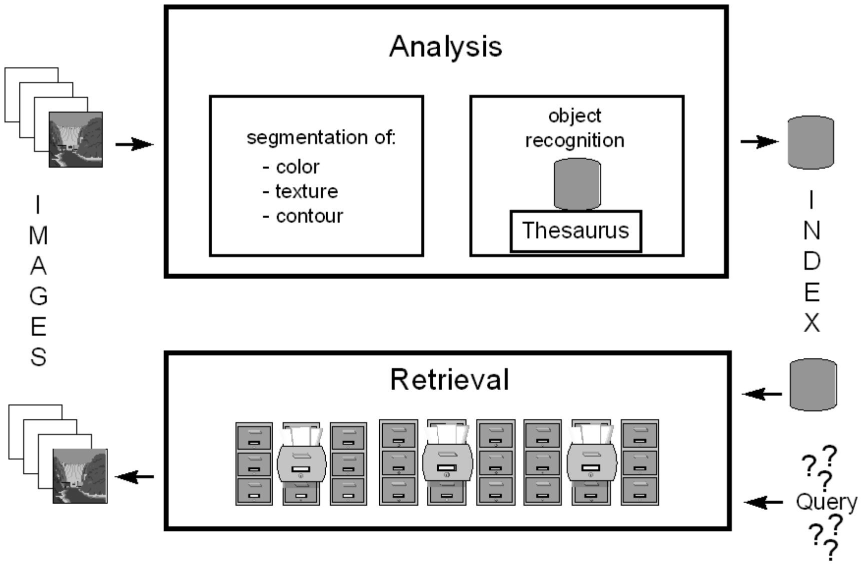


Figure 25.1: *Architecture of the IRIS system.*

creases, because an annotation always depends on the viewer's subjective assessment of an image. The consultation of a thesaurus may help in this respect but will not eliminate the problem completely. The only system (OMNIS) with a fully automatic annotation is restricted to text documents.

25.2 Overview

The IRIS system consists of two main modules: the image analysis module and the retrieval module. The architecture of the IRIS system is presented in Fig. 25.1. The image analysis module consists of four submodules: three modules each extract one of the low-level features, color, texture, and contour. The fourth module implements the *object recognition*. The color module, the texture module, and the contour module extract segments, which are represented as structured texts. These features are extracted independently of each other providing three independent information sources. These sources present the low-level annotations of an analyzed image.

The object-recognition module is based on the generated annotations of the three low-level modules. First, the *neighborhood relations* of the extracted segments are computed. *Graphs* (Volume 2, Chapter 24) are an adequate representation for these neighborhood relations. Second, the object recognition is realized by graph operations triggered by a graph grammar. The graph grammar presents the compiled taxonomy, which reflects the domain knowledge. The object recognition provides the information for the fourth field of the annotation belonging to the image.

The automatic annotation generation is now finished, and the text description is completed. Therefore, the textual description can be indexed by commercially available text retrieval techniques. A query is based exclusively on the text annotations and, thereby, the response time is minimized. Each of the four modules is discussed in further detail in the next sections.

25.2.1 Color

For color-based segmentation we use *color histograms* in the HLS color model (**h**ue, **l**ightness, and **s**aturation, [29]). We compute color histograms for every detected region. The color appearing most frequently defines the color of a region. In the next step, a circumscribing rectangle is determined for every region. Every rectangle has a computed color with respect to the color-naming system (CNS) [30]. The result of the color-based segmentation is called *color rectangles*, with attributes such as size, position, and color.

25.2.2 Texture

The local distribution and variation of the gray values within a region determine the *texture* of the region. Parts of texture regions can be repeated. The texture features of a region are characteristic for the whole region so that there is a coherence of the texture properties within a region. These properties fence off the texture regions. For a more detailed overview of *texture analysis* discussion see Volume 2, Chapter 12.

Conners and Harlow [31] described statistical features as most suitable for analyzing *natural textures*. Therefore, we use the *co-occurrence matrix* as a *texture model*. In Haralick et al. [32] (see also Volume 2, Section 12.2.1) and Gotlieb and Kreyszig [33], co-occurrence features were suggested.

The analysis, classification, and the determination of the textures we use is described in Asendorf and Hermes [34]. First, we subdivide the whole image by an arbitrary, homogeneous grid. For every grid element we calculate the co-occurrence features, for example, angular second moment and variance. We use a neural net as a classifier that maps the co-occurrence features to textures. Grid elements with the same texture are grouped together and the circumscribing rectangle is determined. The results of the texture-based segmentation is called *texture rectangles* with attributes such as size, position, and the classified texture.

25.2.3 Contour

The contour-based shape description in our approach consists of the following three steps: *gradient-based edge detection*; determination of object contours; and *shape analysis*. The shape analysis results in a list of region parameters. They will be given to the module for object recognition. We will only present the main algorithms that have currently been implemented in the IRIS system. For a more detailed overview of edge detection, see Chapter 10.

To detect image edges, we first convolve a gray-value image with two convolution kernels that approximate the first derivation of the Gaussian function. The direction and magnitude of the image intensity gradient can then be computed for each pixel. Once we know the image gradient, the next step is to locate the pixels with the steepest slope along the local gradient direction. According to edge detection theory, these points give the real position of image edges. The method of edge detection in our system was first proposed by Korn [35]. A similar one can be found in Canny [36]. The successful detection of edge points depends on the use of convolution kernels that are suited to the local image gray-value changes. The selection of optimal convolution kernels has a direct influence on the extraction of image structures. This is the so-called *scale-space* problem in computer vision (see Volume 2, Chapter 11). Instead of finding optimal kernels for each point, we try to determine optimal convolution kernels with a different deviation for the whole image in our approach. To realize this, we implemented the edge detection algorithm [35] in a pyramid structure.

Unfortunately, edge detection also provides edge points that are caused by noise. At the same time, this may result in incomplete edge points of objects. Therefore, edge points cannot be directly applied for querying images. They have to be connected to form object or region contours. To do that, we use a contour point-connecting algorithm that is described fully in detail by Zhang [37].

The next step in our approach is to extract shape parameters of the closed regions. In the current implementation, we compute the coordinates of the middle point, the size, and the boundary coordinates of each region. These parameters are then stored in a text file.

25.3 Object recognition

To solve the problem of *knowledge-based object recognition* using syntactical pattern recognition methods, two essential steps are necessary:

1. *Bridge* the gap between lower-level (quantitative) information, that is, the information generated from the methods described in the previous sections, and the atomic entities of the higher-level (qualita-

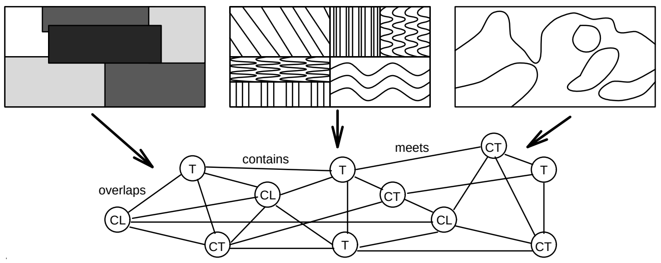


Figure 25.2: *Color (CL), Texture (T) and Contour (CT).*

tive) information, that is, the primitive objects described in a knowledge base. The results of this first step are hypotheses with respect to primitive objects.

2. *Combine* primitive objects according to the *compositional semantics* of more complex objects described in a knowledge base. A hypothesis used in a description of the analyzed image becomes a thesis.

Inherent to the information about color, texture and contour of the image analysis phase is the information about the *topological relations* in the image data between these different segments, as illustrated in Fig. 25.2. These neighborhood relations are distinguished by three cases: *overlaps; meets; and contains* and their inverse relations.

One fundamental assumption of the IRIS system is that these neighborhood relations restrict the search space for the recognition of objects, that is, a (primitive) object is built upon segments that are in these neighborhood relations. By this assumption the process of *object recognition* can be dealt with as a process of *graph transformation*, that is, the process of *graph rewriting*.

In the IRIS system, a graph parser—the so-called *GraPaKL* [38]—is implemented for the *object-recognition* phase. The underlying graph grammar formalism and the parser algorithm are described in Klauck [38, 39]. Within the IRIS system, two graph grammars are used:

1. a grammar to bridge the gap between lower-level information and primitive objects; and
2. a grammar to combine the primitive objects.

The grammars are compiled from the IRIS knowledge base. In this sense, the model of the world recognizable by IRIS is represented within this knowledge base.

25.3.1 Knowledge representation

The knowledge-based tool is an extension and combination of a logic-based *thesaurus* [40], a Knowledge Language ONE (KL-ONE)-like system

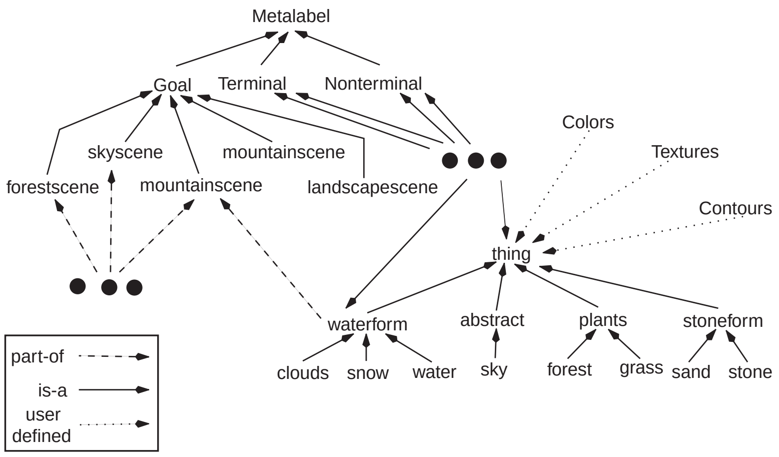


Figure 25.3: *Object definitions.*

[41], and a front-end visualization tool [42]. The main components of this knowledge base—representing the model of an IRIS object world—are the visualization, representation and *consistency checks* components.

The *visualization* component is the graphical user interface of the thesaurus based mainly on *daVinci*, a front-end visualization tool [42]. It offers the user an easy possibility to enter, view and manipulate definitions of objects (Fig. 25.3).

The *representation* component stores the entire knowledge. Several functions are provided for access and modification of the knowledge base. Integrated in this component is a concept language based on KL-ONE [41], and a logic-based thesaurus [40].

This tool offers several *consistency checks* and verifies the defined knowledge for soundness and completeness. This is performed during the *knowledge acquisition*/editing phase. The tests are adapted to the goal to prevent the description of impossible objects. This offers the user the possibility to detect and to eliminate most of the errors as early as possible. Some checks are performed when new or changed knowledge is saved by the user into the knowledge base. The complete check is only performed on request by the user.

25.3.2 Strategies for modeling the domain knowledge

Following the approach of syntactical pattern recognition, a *graph grammar* is a powerful method to handle *object recognition* by substitution of topological graphs. The main point is in finding an adequate and consistent model of the domain. This section concentrates on the underlying modeling strategies of one of the current IRIS grammars that deals with the landscape domain. A larger grammar dealing with the domain of technical drawings contains 105 rules.

The graph grammar consists of three different object types: goal, terminal, and nonterminal (Fig. 25.3). Terminal nodes are represented by the input of the color, texture and contour module. Therefore, the nonterminal nodes are composed of color, texture and contour segments. Hence, it follows that the nonterminal nodes are divided into different object classes: the primitive objects, which are just supported by the color, texture and contour segments (specifying the grammar about the primitive objects); and the complex objects, which are based on the primitive objects.

The strategy for modeling the primitive objects is that they consist always of a color, texture and contour segment. The size of the color and the texture segments are only allowed to differ by a factor of two and both segments are contained by the contour segment. This results in a correspondence of the color and the texture annotation to the same region. In the current grammar, eight primitive objects are modeled where each one is defined by one grammar rule: `sky; clouds; forest; grass; sand; snow; stone;` and `water`.

The current strategies for complex objects models incorporate:

- complex objects that are composed of primitive objects. To reduce the number of rules for a concept definition, a subsumption hierarchy is introduced;
- primitive objects that are generally specified by their size, for example, a complex object of label `forestscene` should be dominated by a large segment of label `forest`; and
- primitive objects are related by the topological relations `meets, contains` and `overlaps` to ensure their neighborhood.

So far, five complex objects have been modeled. The number of rules is notated in parentheses: `landscape scene` (4 rules); `mountain scene` (4 rules); `mountain lake scene` (8 rules); `sky scene` (6 rules); and `forest scene` (6 rules).

The current IRIS grammar is composed only of three layers: segments as terminals (first level); primitive objects as nonterminals (second level); and complex object as goals (third level).

25.4 Examples

In this section, we give a short example from the result of image analysis on the image illustrated in Fig. 25.4a. The original image is in RGB model with the size of 728×472 pixels. Figure 25.4b gives the result of color-based segmentation.

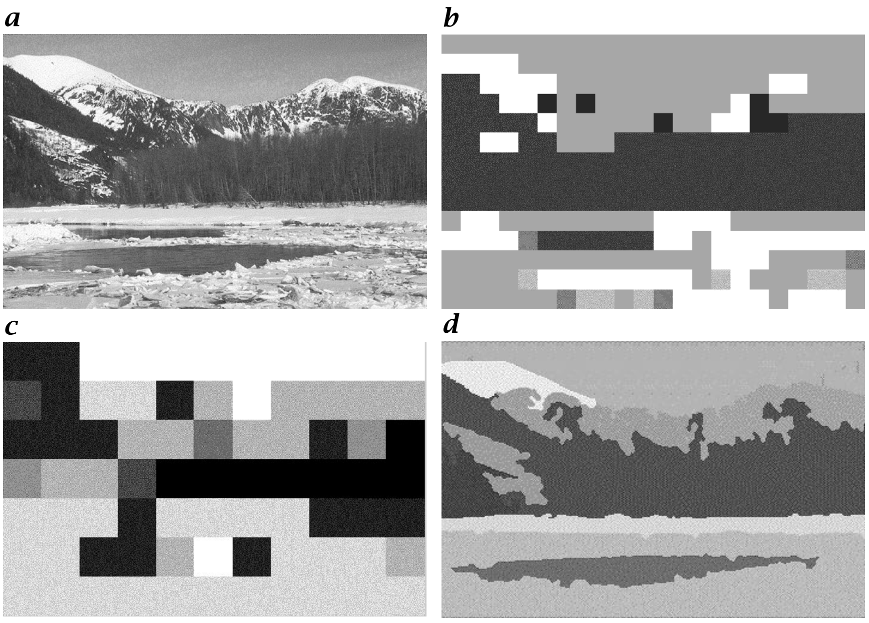


Figure 25.4: ***a*** *Original landscape image;* ***b*** *result of the color-based segmentation;* ***c*** *result of the texture-based segmentation; and* ***d*** *result of the contour-based segmentation.*

The segmentation based on texture is shown in Fig. 25.4c. For the contour-based shape analysis the contours are extracted in form of a contour map. To determine the dominant regions, a region or shape analysis is then carried out (Fig. 25.4d).

After the image-analysis phase, the *object-recognition* phase is started. First, the preprocessing step, then, the generating of the hypothesises, that is, possible primitive objects are recognized in the image. In the third step the primitive objects are combined according to the compositional semantic of objects defined in the thesaurus to more complex objects. The overall description of the image is then given by the one or more resulting object structures as parses. Figure 25.5 presents the main window of the object recognition. In the upper left, a list of the three recognized objects is shown, that is, `forest scene`, `landscape scene` and `sky scene`. The lower left subwindow displays attributes of the chosen object. The structure of the object `forest scene` is illustrated as a parse tree on the right.

See also the demo version of our image retrieval system on the CD-ROM included with this handbook (`/software/25`).

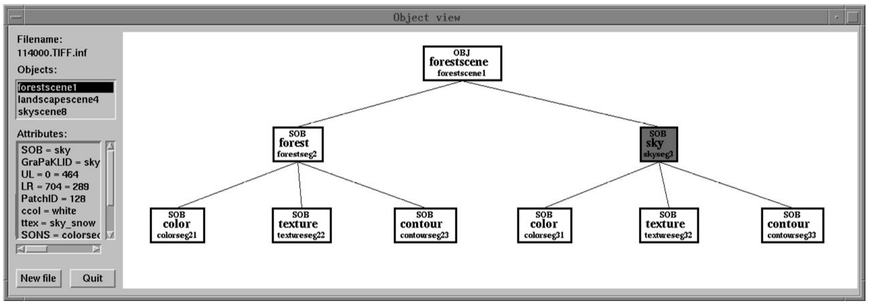

Filename:
114000.TIFF.inf
Objects:
forestscene1
landscapescene4
skyscene8
Attributes:
SOB = sky
GraPaKLID = sky
UL = 0 = 464
LR = 704 = 289
PatchID = 128
ccol = white
ttex = sky_snow
SONS = colorsec

Figure 25.5: *Results of the object recognition.*

25.5 Conclusion

We demonstrated how techniques of computer vision, graph grammar theory, and text retrieval can be combined in a novel way to implement an efficient image retrieval system, which also delivers the basic funtionality to the analysis of video frames.

25.6 References

[1] Alshuth, P., Hermes, T., Klauck, C., Kreyß, J., and Röper, M., (1996). IRIS Image retrieval for images and Videos. In *Proceedings of first Int. Workshop of Image Databases and Multi-Media Search, IDB-MMS, Amsterdam, The Netherlands*, pp. 170–178.

[2] Alshuth, P., Hermes, T., Voigt, L., and Herzog, O., (1998). On video retrieval: content analysis by ImageMiner. In *IS&T/SPIE Symposium on Electronical Imaging Sciene & Technology (Storage and Retrieval for Images and Video Databases), San Jose, CA*, Vol. 3312, pp. 236–247.

[3] Nagy, G., (1985). Image databases. *Image and Vision Computing*, **3**(3): 111–117.

[4] Cawkell, A., (1992). Imaging systems and picture collection management: a review. *Information Services & Use*, **12**:301–325.

[5] Jain, R. (ed.), (1992). *NSF Workshop on Visual Information Management Systems*. Ann Arbor, Mich.: Computer Science and Engineering Division, The University of Michigan: Workshop Report.

[6] Leung, C., (1990). Architecture on an image database system. *Information Services & Use*, **10**:391–397.

[7] Bordogna, G., (1990). Pictorial indexing for an integrated pictorial and textual information retrieval environment. *Information Services & Use*, **16**:165–173.

[8] Chakravarthy, A., (1994). Toward semantic retrieval of pictures and video. In *Proc. Riao'94, Intelligent Multimedia Information Retrieval Systems and Management*, pp. 676–686. New York.

[9] Hermes, T., Klauck, C., Kreyß, J., and Zhang, J., (1995). Image retrieval for information systems. In *Proc. SPIE—The Inter. Soc. for Optical Engineering, Storage and Retrieval for Image and Video Databases*, pp. 394–403.

[10] Brolio, J., Draper, B., Beveridge, J., and Hanson, A., (1989). SR: a database for symbolic processing in computer vision. *Computer*, **22**(12):22–30.

[11] Swain, M. and Ballard, D., (1990). Indexing via color histograms. In *DARPA Image Understanding Workshop, Pittsburgh, Pennsylvania, September 11-13*, pp. 623–630.

[12] Chen, C. and Chang, C., (1993). An object-oriented similarity retrieval algorithm for iconic image databases. *Pattern Recognition Letters*, **14**: 465–470.

[13] Niblack, W., Barber, R., Equitz, W., Flickner, M., Petkovic, D., Yanker, P., Faloutsos, C., and Taubin, G., (1993). The QBIC project: Querying image by content using color, texture, and shape. In *Proc. SPIE Storage and Retrieval for Image and Video Database, San Jose, CA*, pp. 173–181.

[14] Binaghi, E., Gagliardi, I., and Schettini, T., (1994). Image retrieval using fuzzy evaluation of color similarity. *International Jour. Pattern Recognition and Artificial Intelligence*, **8**:945–968.

[15] Pentland, A., Picard, R., and Sclaroff, S., (1994). Photobook: Tools for content-based manipulation of image database. In *Proc. of SPIE on Storage and Retrieval for Image and Video Databases, San Jose, CA*, pp. 34–47.

[16] Picard, R. and Minka, T., (1995). Vision texture for annotation. *Multimedia Systems*, **3**(1):3–14.

[17] Yeung, M., Yeo, B., Wolf, W., and Liu, B., (1995). Video browsing using clustering and scene transitions on compressed sequences. In *IS&T/SPIE Symposium on Electronical Imaging Science & Technology (Multimedia Computing and Networking), San Jose, CA.*

[18] Deardorff, E., Little, T., Marskall, J., Venkatesh, D., and Walzer, R., (1994). Video scene decomposition with the motion picture parser. In *IS&T/SPIE Symposium on Electronical Imaging Science & Technology (Digital Video Compression and Processing on Personal Computers: Algorithms and Technologies), San Jose, CA*, Vol. SPIE 2187, pp. 44–55.

[19] Desai, U. *Coding of Segmented Image Sequences.* M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 286, (1994).

[20] Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., and Yanker, P., (1995). Query by image and video content: the QBIC system. *IEEE Computer: Special issue on Content Based Picture Retrieval Systems.*

[21] Hirata, K. and Kato, T., (1992). Query by visual example. In *Proceedings of Third Intl. Conf. on Extending Database Technology, Viennna, Austria*, pp. 56–71.

[22] Little, T., Ahanger, G., Folz, R., Gibbon, J., Reeve, F., Schelleng, D., and Venkatesh, D., (1993). A digital on-demand video service supporting content-based queries. In *Proceedings of 1st ACM Intl. Conf. on Multimedia, Anaheim, CA*, pp. 427–436.

[23] Rowe, L., Boreczky, J., and Eads, C., (1994). Indexes for user access to large video databases. In *IS&T/SPIE Symposium on Electronical Imaging Science & Technology (Storage and Retrieval for Image and Video Databases II), San Jose, CA*.

[24] Davis, M., (1993). Media stream: An iconic language for video annotation. In *Proceedings of IEEE Symposium on Visual Languages, Bergen, Norway*, pp. 196–202.

[25] Wiesener, S., Kowarschik, W., Vogel, P., and Bayer, R., (1995). Semantic hypermedia retrieval in digital Libraries. In *Proceedings of IEEE Workshop on Advances in Digital Libraries*, pp. 73–85. New York: Springer.

[26] Devadiga, S., Kosiba, D., Oswald, S., and Kasturi, R., (1995). A semiautomatic video database system. In *IS&T/SPIE Conference on Storage and Retrieval for Image and Video Databases III, San Jose, CA*, Vol. 2420, pp. 262–267.

[27] Zhao, D. and Ramsden, A., (1995). The Elinor electronic library. In *Proceedings of IEEE Workshop on Advances in Digital Libraries*, pp. 195–207. New York: Springer.

[28] Hjelsvold, R. and Midtstraum, R., (1995). Databases for video information sharing. In *IS&T/SPIE Conference on Storage and Retrieval for Image and Video Databases III, San Jose, CA*, Vol. 2420, pp. 268–279.

[29] Foley, J., van Damm, A., Feiner, S., and Hughes, J., (1990). *Computer Graphics: Principles and Practice*, 2nd edition. Reading, MA: Addison-Wesley. Revised 5th Printing, 1993.

[30] Berk, T., Brownston, L., and Kaufmann, A., (1982). A new color-naming system for graphics languages. *IEEE CG&A*, **2**(3):37–44.

[31] Conners, R. and Harlow, C., (1980). A theoretical comparison of texture algorithms. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **2**(3): 204–222.

[32] Haralick, R., Shangmugam, K., and Dinstein, I., (1973). Textural features for image classification. *IEEE Trans. Systems, Man, and Cybernetics*, **3**(6): 610–621.

[33] Gotlieb, C. and Kreyszig, H., (1990). Texture descriptors based on co-occurrence matrices. *Computer Vision, Graphics, And Image Processing*, **51**:70–86.

[34] Asendorf, G. and Hermes, T., (1996). On textures: An approach for a new abstract description language. In *Proceedings of the IS&T/SPIE's Symposium on Electronic Imaging 96, 29 January - 1 February*, Vol. 2657, pp. 98–106.

[35] Korn, A., (1988). Toward a symbolic representation of intensity changes in images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **10**:610–625.

[36] Canny, J., (1986). A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **8**(6):679–698.

[37] Zhang, J., (1994). *Region-based road recognition for guiding autonomous vehicles*. PhD thesis, Department of Computer Science, University of

Karlsruhe, Germany, Feb. 1994, VDI Berichte 298, VDI Verlag, Düsseldorf. (in German).

[38] Klauck, C., (1994). *Eine Graphgrammatik zur Repräsentation und Erkennung von Features in CAD/CAM*, Vol. No. 66 of *DISKI*. St. Augustin: infix-Verlag. PhD Thesis, University of Kaiserslautern.

[39] Klauck, C., (1995). Graph grammar based object recognition for image retrieval. In *Proceedings of the 2nd Asian Conference on Computer Vision 1995 (ACCV'95)*, Vol. 2, pp. 539–543.

[40] Goeser, S., (1994). A logic-based approach to thesaurus modeling. In *Proceedings of the International Conference on Intelligent Multimedia Information Retrieval Systems and Management (RIAO) 94*, pp. 185–196. C.I.D.-C.A.S.I.S.

[41] Hanschke, P., Abecker, A., and Drollinger, D., (1991). TAXON: A concept language with concrete domains. In *Proceedings of the International Conference on Processing Declarative Knowledge (PDK) 91*, pp. 411–413. Berlin: Springer, LNAI 567.

[42] Fröhlich, M. and Werner, M., (1994). Demonstration of the interactive graph visualization system daVinci. In *Proceedings of DIMACS Workshop on Graph Drawing '94*, LNCS 894, pp. 266–269. Berlin: Springer.

26 A Tactile Vision-Substitution System

Markus Loose, Thorsten Maucher, Johannes Schemmel, Karlheinz Meier, and Michael Keller

Institut für Hochenergiephysik (IHEP), Universität Heidelberg, Germany

26.1 Introduction

An early discussion of sensory substitution can be found in the book Cybernetics by Norbert Wiener [1]. Vision substitution systems with tactile displays in particular have been successfully employed for the first time about 30 years ago by Bach-y-Rita and co-workers [2]. The authors of those early experimental studies claim that after sufficient training blind subjects were able to perceive visual images in space using their *somatosensory system*. Despite this principal success practical applications have been hindered by the fact that the two basic components, image acquisition and *tactile display* were rather clumsy thus excluding portable devices. In the meantime technology has advanced in particular in the areas of integrated *microelectronics* and high performance computing. The first area allows to design highly compact *photodetector arrays* with specifications similar to the front-end of the human visual system. The second area makes it possible to read out and process digital data in real time. Using these technologies together with a novel concept of a *virtual tactile display* we have developed, built and tested a *tactile vision-substitution system* (TVSS). The system

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

ISBN 0–12–379773-X/$30.00

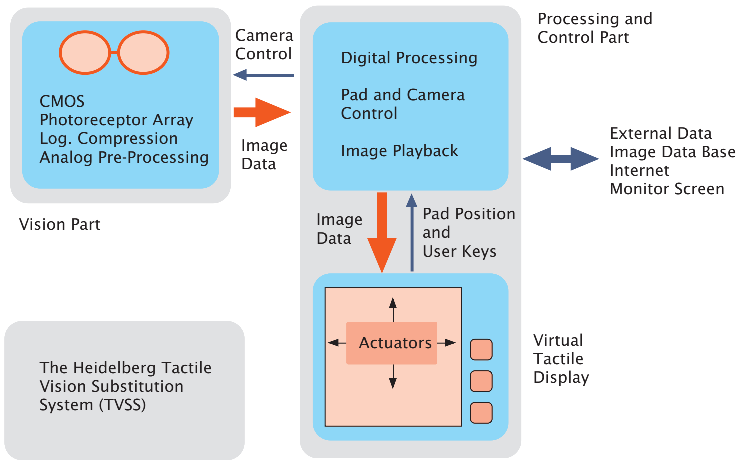


Figure 26.1: *Schematic overview of the complete tactile vision-substitution system (TVSS).*

gives blind people access to the real visual world as well as to computer graphics, drawings and similar artificial patterns.

The complete system is fully operational as a prototype. Testing is carried out in a joint research project with the eye clinic at Heidelberg University.

26.2 Concept and realization

The TVSS presented here consists of a vision part responsible for acquisition and preprocessing of visual information, a processing and control part for data acquisition, further image processing and control of the connected hardware devices and a novel type of virtual tactile display based on conventional Braille code cells mounted on an opto-mechanical system for position-sensitive output of the processed image information. The acquired and processed image can be explored by the user on the tactile display. Figure 26.1 shows a schematic overview of the complete system.

26.2.1 The vision part

The vision part is a single chip camera system designed and built using standard complementary metal oxide semiconductor (CMOS) technology. The availability of basic circuit elements such as diodes and transistors suggests the use of the intrinsic photoelectric effect in the depletion zone of CMOS pn-structures for light detection. At the same time, the possibilities for further signal processing on the same silicon wafer can be used. This is the basic idea of CMOS photoreceptors *vision chips* (see also Chapter 7). Apart from the integration of sensor elements and signal processing CMOS photoreceptors are characterized by a huge dynamic range of 1 : 1 million or more. This performance comes close to the biological model and is far better than that of conventional charge coupled device (CCD) systems. This is of great advantage for the application in mind, because such a system can operate under a large range of different illuminations without the need for additional aperture corrections that would require additional space and electrical power. In practical applications, however, the use of the large dynamic range is difficult. A linear transfer of many orders of magnitude in incident illumination requires an unrealistic range of output voltages. A solution to this problem also realized in biological systems is a *logarithmic compression* of the electrical output signal. Examples for an electronic implementation can be found in Chapters 8 and 9. Two types of logarithmic response camera systems with different specifications have been produced in the *application-specific integrated circuit* (ASIC) laboratory at the Faculty of Physics at Heidelberg University.

The first system type is based on the 1.2 μm CMOS technology of Austria Micro Systems (AMS), Unterpremstätten, Austria. This CMOS process offers two metal and two polysilicon layers. The system uses the *adaptive pixel* concept described by Delbrueck and Mead [3] with logarithmic response to the incident light intensity, and has been developed in the initial phase of the Heidelberg vision project. Detailed measurements performed with this camera system describing, in particular, the phototransduction mechanism characterizing the specific CMOS process can be found in Loose [4], Loose et al. [5, 6, 7], and Schemmel [8].

All CMOS vision chips have to cope with the problem of mismatch between individual pixels. This well-known effect is called *fixed-pattern noise* and poses a considerable problem for subsequent image processing because it generates artificial local contrasts that are easily detected by edge detection filters. The consequence for the tactile output is severe because the limited bandwidth of the somatosensory system would be overloaded with faulty information. The fixed-pattern noise is a feature aggravated in logarithmic CMOS-based image sensors and currently represents one of their major disadvantages. The reason for this spatial noise can be allocated in the mismatch of individual transistors used for analog preprocessing of pixel data. In principle, the effect can be reduced by increasing the size of the relevant transistors. However, this would increase the geometrical size of the individual pixels and in turn the size of the entire sensor above acceptable limits. The only practical way to handle fixed-pattern noise is to apply a calibration procedure. Measurements carried out with logarithmic pixels in the framework of this project (see references cited

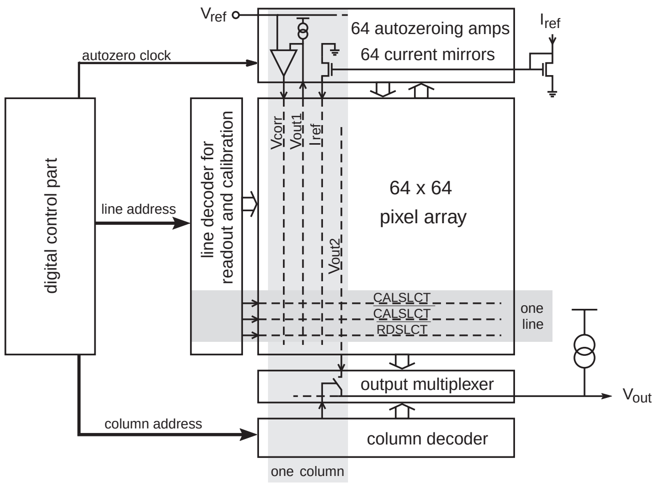


***Figure 26.2:** Architecture of the self-calibrating* 64×64*-pixel vision chip.*

in the foregoing) show that the fixed-pattern noise is dominated by offsets of the logarithmic response curves that can correspond to as much as two to three orders of magnitude in incident light intensity. The pixel-to-pixel variations of logarithmic slopes are usually negligible. Various methods for offset calibration have been proposed and implemented. The most obvious way is to use software methods. The required post-production calibration procedure and the need for external components and databases are disadvantages that do partly affect some of the major advantages of CMOS sensors, their compactness and low price. On-Chip solutions such as floating-gate technologies or hot carrier degradation as described by Ricquier and Dierickx [9] also need post-production calibration procedures that, in addition, can take several hours for one chip.

Motivated by the need for a logarithmic sensor with small *fixed-pattern noise* the Heidelberg group has developed a vision chip with a built-in analog fixed-pattern noise correction. This chip is based on a *continuously working photoreceptor* with logarithmic response covering a dynamic range of more than 6 decades in incident light intensity. To calibrate the chip the system has to be brought into a reference state that corresponds to a defined sensor input. For that purpose all photoreceptors can be stimulated by the same reference current in a dedicated calibration cycle. Without fixed-pattern noise, all pixel outputs would show the same output signal. In reality, the signals differ due to the problem described in the foregoing. A differential amplifier compares individual outputs to the reference and adjusts each pixel until both signals are equal. The calibration is then stored in an *analog*

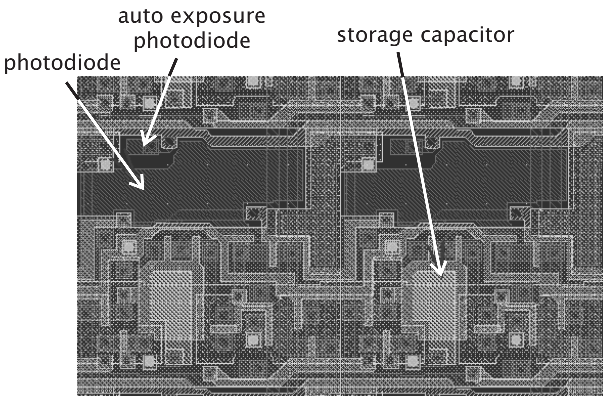


Figure 26.3: *Layout of two pixels in the self-calibrating vision chip. Image-acquisition diodes, exposure-control diodes and storage capacitors are indicated; (see also Plate 4).*

memory available for each individual photoreceptor. Subsequent lines of pixels are being calibrated so that one differential amplifier per pixel column has to be implemented. Because CMOS operational amplifiers show offset variations the autozeroing techniques have been applied for their realization on the chip. The prototype chip features 4096 pixels arranged in a matrix of 64 rows and 64 columns. The architecture of the chip is shown in Fig. 26.2.

As an additional feature the reference source can be replaced by a second array of 64×64 small photodiodes interleaved with the ones acquiring the actual image. The photocurrents of the small diodes are averaged to provide a current representing the average illumination on the chip surface. Using this current as a reference results in an automatic *exposure control* continuously adapting the output signal to the average light intensity.

The chip was fabricated in the 0.8 μm CMOS technology of Austrian Micro Systems. This CMOS process offers two metal layers and two polysilicon layers similar to the one described in the foregoing. Figure 26.3 shows the layout of two pixels. Image-acquisition diodes, exposure-control diodes and storage capacitors are indicated.

One such pixel has an area of $33\,\mu\text{m} \times 33\,\mu\text{m}$. The image-acquisition photodiode itself occupies 20 % of that area. This fill factor could be improved somewhat by going to a CMOS process with a smaller feature size. The layout of the complete vision chip is shown in Fig. 26.4. The digital control part made of standard cells and the quadratic pixel matrix with decoder and amplifier arrays can be seen. In total, the chip has an area of $3.5\,\text{mm} \times 2.5\,\text{mm}$.

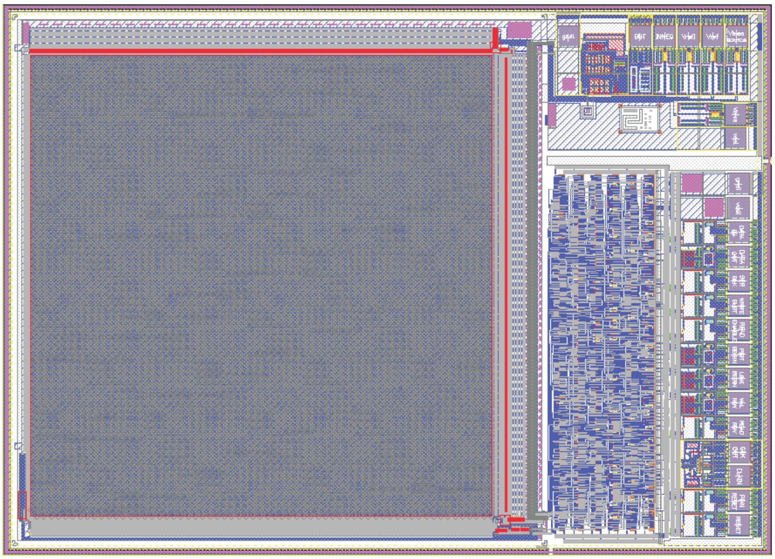


Figure 26.4: *Layout of the complete vision chip. The digital control part made of standard cells and the quadratic pixel matrix with decoder and amplifier arrays can be seen; (see also Plate 5).*

Measurements carried out with the produced chip have confirmed the expected behavior. The chip can safely operate over six orders of magnitude in incident illumination with an almost perfect logarithmic response. Figure 26.5 shows the measured response curve for a pixel on the chip. The slope is negative with an absolute value of 110 mV per decade. This large slope has been achieved by using two serially connected transistors operating in their subthreshold region. The turnover of the response curve for very large light intensities (exceeding the value of the solar constant) can be explained by the discharge of the correction capacitor due to parasitic photocurrents. The saturation of response at very low intensities (below typical moonlight illumination) is caused by the long time constant for charging parasitic capacitances and leakage currents.

The key feature of the chip is the *analog self-calibration* mechanism. At a given uniform illumination the fixed-pattern noise has a standard deviation of about 2 mV, corresponding to a few percent of one decade in illumination. This has to be compared to the typical performance of uncalibrated CMOS vision sensors where fixed-pattern noise corresponding to 2 to 3 decades in illumination has been observed, which makes external calibration mandatory. Figure 26.6 shows the pixel-to-pixel fixed-pattern noise of a 64-pixel column in the produced vision chip. For comparison an uncalibrated reference column realized in the

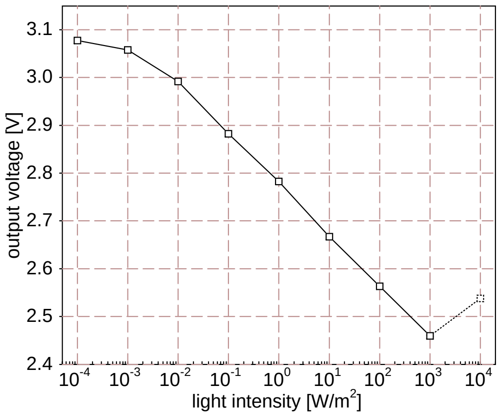


Figure 26.5: *Pixel response measured over 8 decades of incident light intensity.*

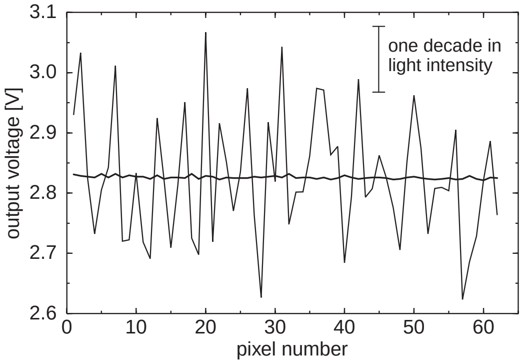


Figure 26.6: *Comparison of a calibrated pixel column in the vision chip (thick line) and a column of uncalibrated test pixels in the Austrian Micro Systems* 0.6 *µm CMOS technology (thin line).*

Austrian Micro Systems 0.6 µm CMOS technology is shown. Details on the self-calibrating CMOS vision chip can be found in [10].

The vision chip has been mounted in a housing and equipped with a simple optics (single lens with a diameter of 6 mm and a focal length of 5 mm). Multiplexed analog image data are sent through a cable to a unit containing the processing and control part as well as the virtual tactile display.

26.2.2 The processing and control part

Data acquisition, further image processing and control of the camera and the tactile display are carried out with a conventional laptop computer mounted in the base of the unit that also serves as a mechanical

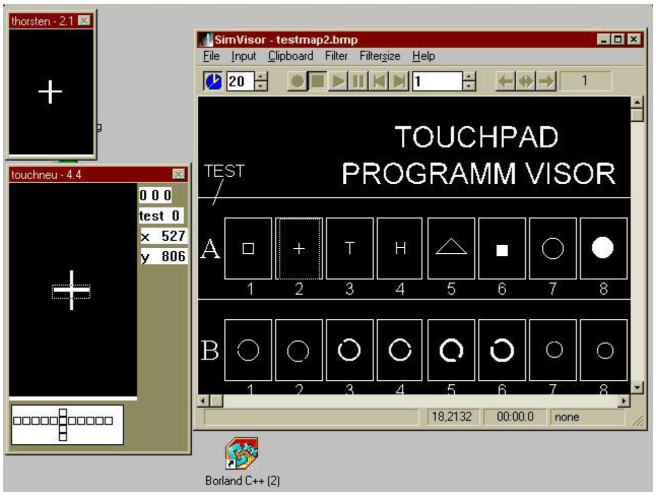


Figure 26.7: Screenshot of the PC running the VISOR program. The displays show computer-generated patterns. The left-bottom window indicates the area displayed by the piezoelectric actuators (dotted rectangle) as well as their actual pattern (square dots).

support for the virtual tactile display described in the next chapter. The complete system has dimensions of 23 cm × 30 cm × 7 cm and a weight of approximately 3 kg determined essentially by the laptop PC.

Multiplexed analog image data from the 4096 camera pixels is transmitted to a receiver board in the display unit. Digitization is performed on this board using an 8-bit ADC. The digitized data are read into the parallel port of the PC. The PC runs a program package called VISOR that has been developed in the framework of this project. VISOR can handle image data from the receiving board or any other image data format from mass storage or even via network. The image data can be further processed with conventional filters. Examples are Sobel filters (see Sections 33.4.2 and 10.3.4) for edge detection or noise-reduction methods. Processed image data are then sent by a PCMCIA digital I/O card to the virtual tactile display. When used as a tactile vision substitution system the PC or an external screen can be used to monitor the performance of the person using the tactile display. Figure 26.7 shows a screenshot of the VISOR software operating with computer-generated patterns from a standard drawing program.

26.2.3 The virtual tactile display

The basic idea behind the virtual tactile display developed in the framework of this project is to limit the actual tactile output with physical

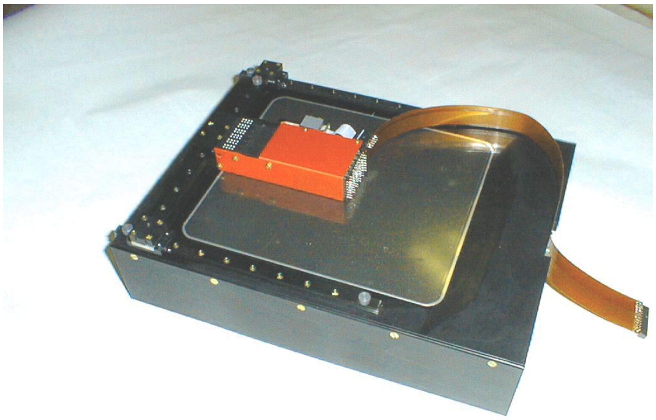


Figure 26.8: *Photograph of the virtual tactile display. The movable matrix of piezoelectric actuators mounted on a system of x-y guiding rails can be seen.*

actuators to a small area accessible with the fingertip. This area represents only a small fraction such as 2 % of the full field of tactile vision. This full field is virtually available but is only displayed once the user decides to move his or her finger to a position of choice. This method represents an economical way to realize a tactile display with a few thousand virtual actuators. Alternative methods such as arrays of tiny magnetic solenoids or electrotactile stimulators require a much larger technological effort.

The physical tactile display unit is based on commercially available *Braille code* cells (Type B-8 available through METEC GmbH, Stuttgart, Germany). A single Braille cell has eight plastic dots that are individually driven by piezoelectric crystal pairs of the bending type. The dot spacing is 2.45 mm and the dot stroke is 0.7 mm. Six such modules each with eight movable *piezoelectrical actuators* are mounted together to form a touchpad with a dimension of roughly 1 cm by 4 cm and a total of 48 actuator dots. This touchpad can be moved on mechanical guiding rails covering a total surface corresponding to 2760 virtual dots. The actual position of the touchpad with the six Braille cells is continuously recorded by an optical tracking system and sent to the laptop computer where it is correlated with the image data. The part of the image data in the acceptance region of the touchpad is then sent back to the pad where it is displayed by the individual actuator dots. The user can explore the entire image by moving the pad over a surface of approximately 12 cm by 18 cm. Only binary (i. e., on/off) information can be displayed. The system can operate with a frame rate of 10 to

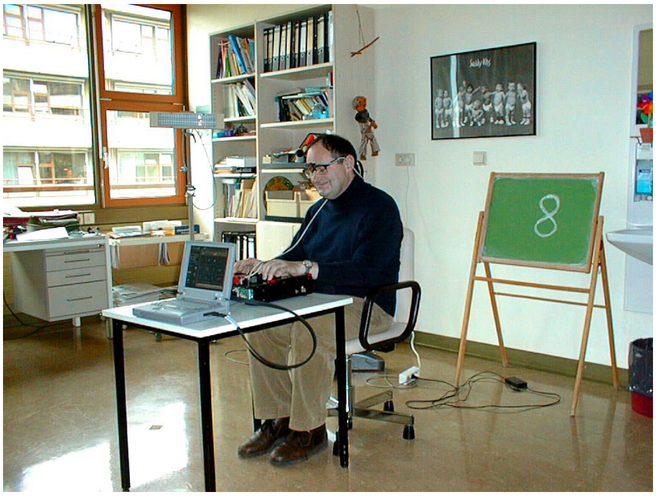


Figure 26.9: *User of the TVSS carrying the vision part and exploring the virtual display.*

20 Hz depending on the amount of image processing carried out on the laptop PC. Figure 26.8 shows a photograph of the virtual tactile display unit. In Fig. 26.7 (left-bottom window, quadratic dot pattern) the positions of the actuator dots representing the actual tactile display pattern are shown for the case of a computer-generated image.

26.3 Results and prospects

The complete system has been tested in the eye clinic at Heidelberg University. A blind person has been trained for a few hours to use the device. Performance studies have been carried out with simple geometrical patterns as well as with real-world scenes recorded by the camera. Figure 26.9 shows a photograph of the person carrying the camera mounted on eye glasses and exploring the tactile display. Further work will be carried out to investigate the possibilities and limitations of this kind of sensory substitution.

Acknowledgments

This work is supported by the Ministerium für Wissenschaft, Forschung und Kunst, Baden-Württemberg, Stuttgart, Germany

26.4 References

[1] Wiener, N., (1948). *Cybernetics or Control and Communication in the Animal and the Machine, Chapter VI.* Cambridge, MA: Massachusetts Institute of Technology.

[2] Bach-y Rita, P., Collins, C., Saunders, F., White, F., and Scadden, L., (1969). Vision substitution by tactile image projection. *Nature*, **221**:963.

[3] Delbrueck, T. and Mead, C., (1995). Analog VLSI phototransduction by continuous, adaptive, logarithmic photoreceptor circuits. Computation and neural systems program, Memo No. 30. In *Vision Chips, Implementing vision algorithms with analog VLSI circuits, Chapter 2*, C. Koch and H. Li, eds., pp. 139–161. Los Alamitos, CA: IEEE Computer Society Press.

[4] Loose, M., (1996). *Layout und Test eines Systems adaptiver Photorezeptoren in analoger CMOS-Technologie.* Diploma Thesis, in German, Heidelberg University, Heidelberg, Germany.

[5] Loose, M., Meier, K., and Schemmel, J., (1996). A camera with adaptive photoreceptors for a tactile vision aid. In *Intelligent Robots and Computer Vision XV: Algorithms, Techniques, Active Vision and Materials Handling, Proc. SPIE*, D. Casaent, ed., Vol. 2904, pp. 528–537.

[6] Loose, M., Meier, K., and Schemmel, J., (1996). A camera with adaptive photoreceptors in analog CMOS technology. In *International Symposium on Advanced Imaging and Network Technologies—Advanced Focal Plane Arrays and Electronic Camera, Proc. SPIE*, T. Bernard, ed., Vol. 2950, pp. 121–130. SPIE.

[7] Loose, M., Meier, K., and Schemmel, J., (1996). Entwicklung einer Kamera mit adaptiven Photorezeptoren in analoger CMOS Technologie. In *Mustererkennung 1996, Proc. 18. DAGM Symposium*, B. Jähne, P. Geißler, H. Haußecker, and F. Hering, eds., pp. 301–312. Berlin: Springer.

[8] Schemmel, J., (1996). *Design und Test einer Single-Chip Kamera mit integrierter Analog/Digital-Wandlung auf der Basis adaptiver Photorezeptoren.* Diploma Thesis, in German, Heidelberg University, Heidelberg, Germany.

[9] Ricquier, N. and Dierickx, B., (1995). Active pixel CMOS image sensor with on-chip non-uniformity correction. In *IEEE Workshop on CCD and Advanced Image Sensors, California, April 20-22,1995.*

[10] Loose, M., Meier, K., and Schemmel, J., (1998). CMOS image sensor with logarithmic response and a self calibrating fixed pattern noise correction, submitted. In *International Symposium on Advanced Imaging and Network Technologies—Advanced Focal Plane Arrays and electronic Cameras, Zürich, Switzerland, 1998*, T. Bernard, ed. SPIE.

27 The Neural Active Vision System NAVIS

Bärbel Mertsching, Maik Bollmann, Rainer Hoischen, and Steffen Schmalz

AG Informatikmethoden für Mikroelektronikanwendungen
Universität Hamburg, Germany

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

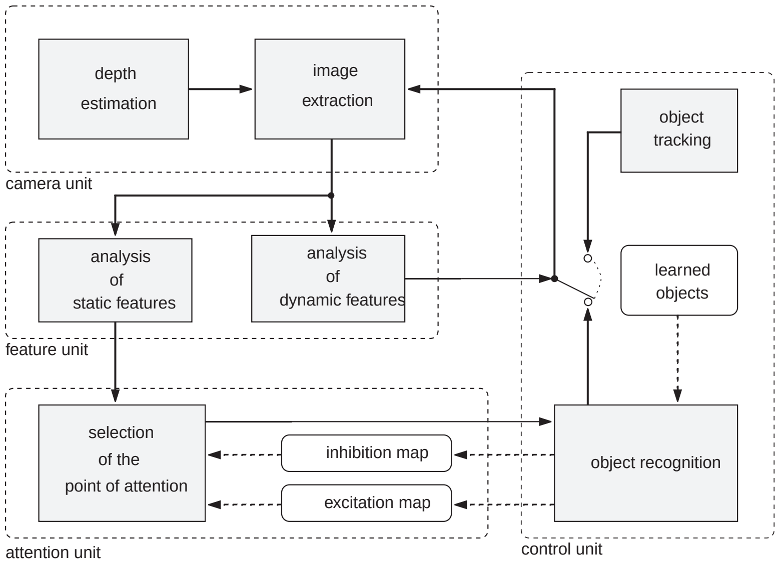


Figure 27.1: *NAVIS architecture.*

27.1 Introduction

In general, active vision systems are characterized by the ability to explore their environment autonomously, to gather the information relevant to the current task, and to react in a flexible way to unexpected events. Such a complex behavior can only be performed by a modular system interacting on several hierarchical levels as already proposed by Granlund [1].

This chapter introduces the *N*eural *A*ctive *V*ision *S*ystem NAVIS that has been developed by the authors in cooperation with the GET Vision Lab, University of Paderborn. Emphasis has been put on the system's biological plausibility as well as on its computational costs and the integration aspects. The integration of several processing units is schematically depicted in Fig. 27.1.

The system can be coupled with two different experimental platforms: a stereo camera head and a mobile robot with a smaller monocular camera head (see Section 27.2). The stereo camera head enables NAVIS to gather depth information from a scene while the mobile robot is able to navigate through the scene. Images that are captured at defined states of the system are analyzed with several static and dynamic features considered. These features are stored in maps and evaluated for salient structures. The attention unit selects the next fixation point in dependence on the current task given by a superior instance. This instance determines whether a static object is searched for and recognized or whether an object in motion is pursued. The corresponding recognition or tracking model guides the camera unit (Section 27.5). The basic principle of the 2-D recognition system is the match between object parts on the basis of rigid point configurations determined by the attention unit. The more flexible 3-D object recognition combines viewer-centered representations with temporal associations reducing the amount of misclassification (Section 27.6). Furthermore, the system must be able to respond to motion stimuli initiated by moving patterns. Therefore, a tracking system responsible for the estimation of motion and pursuit of moving objects is also part of NAVIS (Section 27.7).

27.2 Experimental platforms

The NAVIS system is equipped mainly with two devices for visual scene exploration: A binocular camera head and a mobile robot supplied with a single camera. The camera head as the major vision system delivers a stereo image pair, which can be used for active scene exploration, object recognition, tracking and depth estimation. Furthermore, it could be used to guide and survey the mobile robot as an advanced system to perform action-perception cycles, for example, in conjunction with a hand-eye calibration.

27.2.1 The NAVIS camera head

The primary visual front-end consists of a stereo camera head with two color cameras. It was developed at the University of Paderborn with special consideration given to enhanced calibration capabilities and low costs of realization using off-the-shelf components [2].

The camera modules rotate around independent vertical vergence axes and are mounted on a common pan/tilt platform. This design provides four mechanical degrees of freedom (DoF) for the simulation of eye and neck movements. Figure 27.2 shows an outline sketch of the frontal and side view of the camera head. In Fig. 27.3a the camera head is displayed.

Each camera module also has three translational DoF for its attachment to the platform. This capability extends the manual adjustment facilities for calibration purposes in order to place a camera's optical center in the intersection point of the pan and vergence axis. The camera modules internally provide three additional adjustment facilities. Each camera block can be rotated around the optical axis to align both cameras in the optical plane. In addition, each camera's pitch and yaw angle related to the module housing can be varied over a small range in order to place the sensor plane perpendicularly to the optical axis.

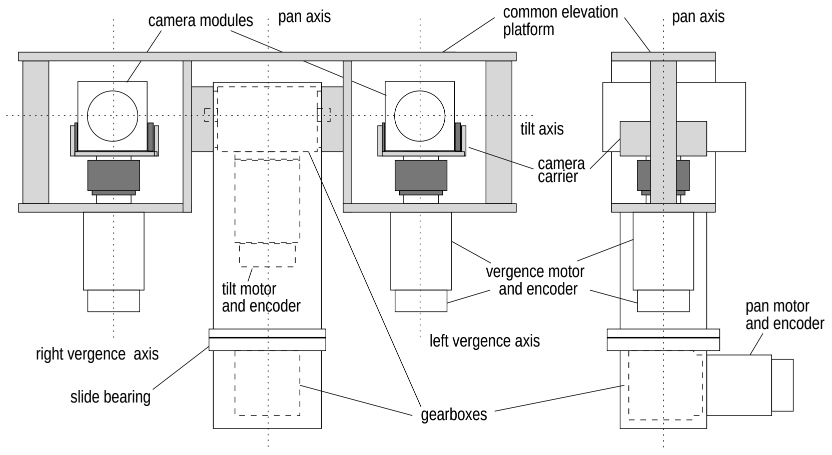


***Figure 27.2:** Outline sketch of the NAVIS camera head. The common elevation platform is marked in light gray. Parts in dark gray enable manual adjustments of the modules for calibration purposes.*

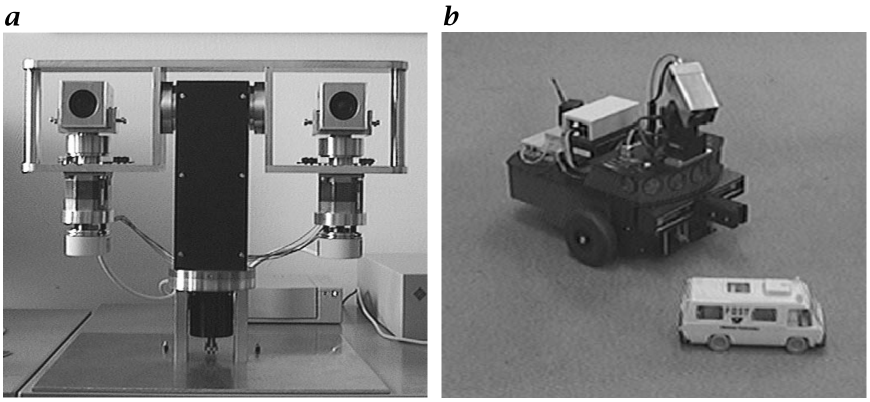


***Figure 27.3:** Experimental platforms: **a** stereo camera head; **b** mobile robot.*

27.2.2 The Pioneer robot

A Pioneer 1 robot [3] serves as the mobile experimental platform. The basic robot consists of a differential wheel drive, seven sonars, an onboard controller, and a one DoF gripper. It was extended to meet special needs for an active vision environment: An independent pan/tilt unit was mounted on top of the robot together with a single color camera. Furthermore, a radio data connection had to be established for the video data as well as for serial command data for the robot and the pan/tilt unit. Thus a complete autonomous system was built up (see Fig. 27.3b).

27.3 Image preprocessing

This section deals with the image preprocessing mechanisms provided in the feature unit of NAVIS.

27.3.1 Edge extraction

Input images are filtered by sets of oriented *Gabor filters* with mean spatial frequency vectors $\tilde{\boldsymbol{k}}$ normalized to the Nyquist wave number (see Eq. (2.34) in Volume 2, Section 2.4.2). Each set is based upon a frequency vector of constant norm $\| \tilde{\boldsymbol{k}} \| = \tilde{k}_0$. It was shown by Jones and Palmer [4] that this filter type offers an appropriate model for certain early vision processes in the human visual system.

The Gabor filters (Volume 2, Section 4.2.2) are given by

$$g(\boldsymbol{x}) = \frac{1}{2\pi\sigma_x\sigma_y}\exp(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x})\exp(\mathrm{i}\tilde{\boldsymbol{k}}^T\boldsymbol{x}) \tag{27.1}$$

where $\boldsymbol{A}$ is composed of a diagonal parameter matrix $\boldsymbol{P}$ and a rotation matrix $\boldsymbol{R}$

$$\boldsymbol{A} = \boldsymbol{R}\boldsymbol{P}\boldsymbol{R}^T = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} \sigma_x^{-2} & 0 \\ 0 & \sigma_y^{-2} \end{bmatrix} \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} \tag{27.2}$$

Setting $\tilde{\boldsymbol{k}}$ to $\left[\tilde{k}_0\cos\varphi, \tilde{k}_0\sin\varphi\right]^T$, $\varphi = n\Delta\varphi$ results in a set of circularly arranged filters in frequency space. In addition, appropriate values for the parameters σ_x and σ_y have to be chosen to adjust the filter bandwidth in radial and tangential directions. To create filter sets of different scales, the norm of the mean frequency k_0 can also be modified, for example, in a logarithmic scale. If the parameters σ_x and σ_y are adopted in an appropriate manner a filter set for different scales and orientations is obtained.

So far, many design concepts for this type of filter set have been introduced, for example, by Theimer and Mallot [5]. For a detailed description of the design of the filter set used in conjunction with NAVIS, see Trapp [6].

27.3.2 Gray-level segmentation

A gray-level segmentation is performed in NAVIS on the basis of gradient images. These gradient images are the result of a filter operation with Sobel kernels (horizontal and vertical axes). The segmentation itself is a *region-growing* procedure. It is followed by a step that removes too small segments. Subsequently, neighboring segments are merged during several dilation cycles.

The resulting segments are sorted with regard to their orientation by a *principal component analysis*. The moments of a segment S are given by [7]

$$m_{p,q} = \frac{1}{N} \sum_{(x,y)\in S} (x - \langle x \rangle)^p (y - \langle y \rangle)^q \tag{27.3}$$

With the foregoing expression the covariance matrix $\boldsymbol{C}$ can be described as

$$\boldsymbol{C} = \begin{bmatrix} m_{2,0} & m_{1,1} \\ m_{1,1} & m_{0,2} \end{bmatrix} \tag{27.4}$$

The eigenvector of the largest eigenvalue of the covariance matrix $\boldsymbol{C}$ specifies the orientation of the corresponding segment [7].

27.3.3 Color quantization

In NAVIS color is processed in the *mathematical transform to Munsell* (MTM) color space that is similar to the Munsell renotation system (see [8]). The MTM color space is chosen because of its perceptual uniformity. The RGB to MTM transformation is based on the correction of the Adams color system. Starting point of the transformation is the calculation of the XYZ tristimulus values from RGB:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.608 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.144 \\ 0.000 & 0.066 & 1.112 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}_{NTSC} \tag{27.5}$$

For the Adams color system (M_1, M_2, M_3) follows:

$$\begin{aligned} M_1 &= V(X_c) - V(Y)\ ,\ X_c = 1.020\,X \\ M_2 &= 0.4(V(Z_c) - V(Y))\ ,\ Z_c = 0.847\,Z \\ M_3 &= 0.23V(Y) \end{aligned} \tag{27.6}$$

Here, $V(a) = 11.6a^{1/3} - 1.6$ denotes the nonlinearity of the human visual system. The MTM space (S_1, S_2, L) is derived from an empirically determined correction of the Adams color system

$$\begin{aligned} S_1 &= (8.880 + 0.966\cos\varphi)M_1 \\ S_2 &= (8.025 + 2.558\cos\varphi)M_2 \\ L &= M_3 \end{aligned} \tag{27.7}$$

with $\varphi = \operatorname{atan}(M1/M2)$. The color quantization is achieved by dividing the hue component of the *hue, value, and chroma* (HVC) color space (polar form of the MTM color space) in equidistant angles.

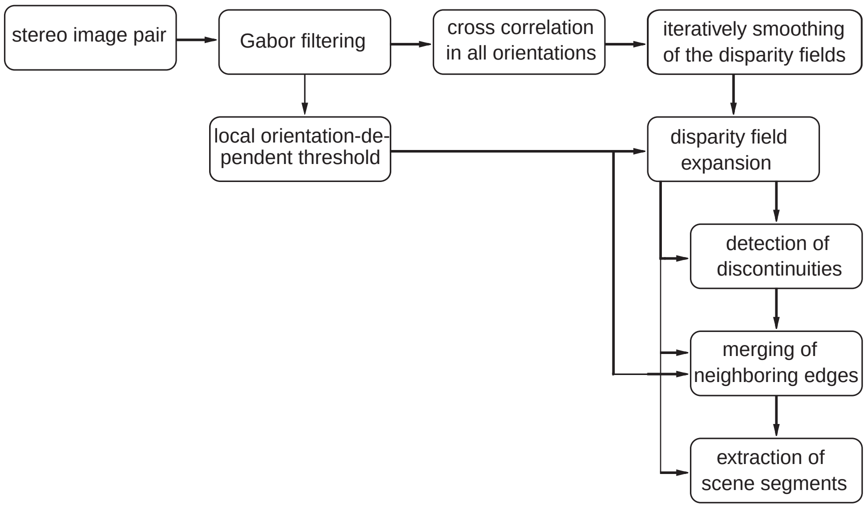

stereo image pair
Gabor filtering
cross correlation in all orientations
iteratively smoothing of the disparity fields
local orientation-dependent threshold
disparity field expansion
detection of discontinuities
merging of neighboring edges
extraction of scene segments

Figure 27.4: *Major steps to form a disparity image and to segment out regions according to their depth.*

27.4 Depth estimation

Binocular visual devices such as the NAVIS camera head (Section 27.2.1) enable the construction of a depth representation of the environment (see also Chapter 17). By comparing the two images obtained from the left or right camera, respectively, one can divide the scene into regions with equal camera distances. Therefore, a figure ground separation for a stable object recognition is possible.

Generating depth data. Depth data (disparity values) are generated using contour segments in different orientations stemming from Gabor filtering (see Section 27.3; [9]). The relation of the left or right image, respectively, is derived by a cross correlation. A single disparity field is determined by applying a continuity constraint. Figure 27.4 shows the major steps to form a disparity image and to segment a scene according to depth values.

The resulting depth maps are sparse (see Fig. 27.5c) due to the randomness of depth assignment. No disparity value is generated if there are any ambiguous depth values for a specific location. Hence, the disparity maps are expanded depending on the extracted contour segments and several criteria:

- contour elements represent borders for the expansion of depth values because of the high probability of a depth discontinuities at these locations;

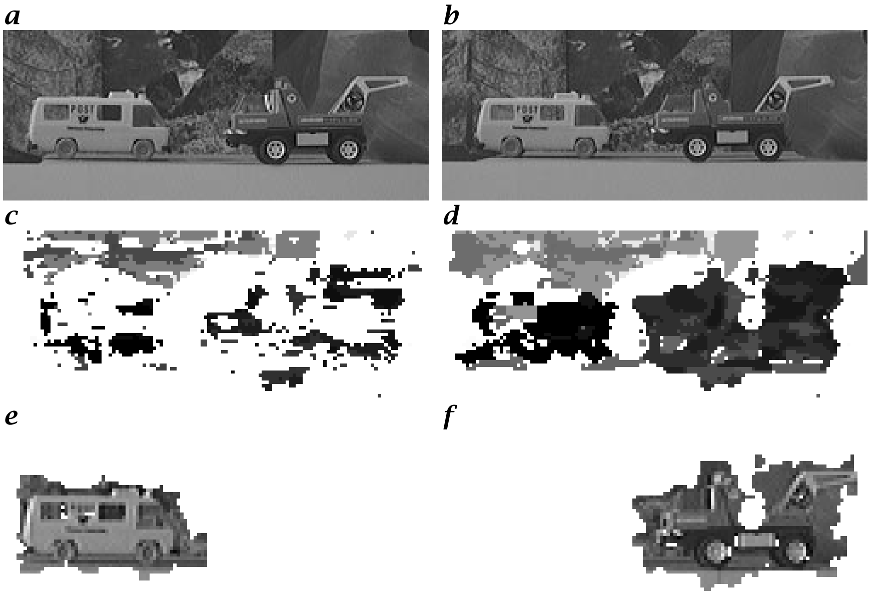


Figure 27.5: *Exemplary depth segmentation:* ***a*** *left picture of a stereo image pair;* ***b*** *right picture;* ***c*** *derived disparity map;* ***d*** *improved disparity map;* ***e*** *and* ***f*** *extracted scene segments (enlarged).*

- if adjacent pixels own different depth values these regions also build borders; and
- depth values mainly spread out along contours. Thus, small gaps between contour segments can be filled.

During an iterative process new depth values are assigned to previously undefined regions (according to their depth). The process stops if there are no new assignments.

Scene segmentation. The expanded disparity map is used for a depth segmentation (see Fig. 27.4) and is applied for a figure-ground separation. First, scene contour elements and depth discontinuities are merged. Points nearest to depth discontinuities are marked and added to the resulting contour elements that are the basis for the scene segmentation. This segmentation is carried out by a simple flood algorithm. Figure 27.5 shows an exemplary scene with two toy cars in different camera distances. After expanding the sparse disparity maps scene regions containing the cars are extracted.

Problems arise due to improper disparity values. An algorithm for generating depth values with subpixel accuracy should allow a more precise segmentation of the scene. A more detailed outline of the algorithms is described in [10].

27.5 Visual attention

27.5.1 Principal ideas

The NAVIS attention module offers a solution to how salient points can be found in real-world scenes. It can be divided into three fundamental components:

- data-driven control;
- model-driven control; and
- behavior.

The data-driven part generates an attractivity representation from the feature structures of the current image section (see Section 27.5.2). In the model-driven part the system follows the instructions inherent in the currently selected model (Section 27.5.3). This model could be one of the following:

- hypothesis generation mode: an object hypothesis is generated depending on recognized configurations of attractivity points;
- hypothesis validation mode: saccadic shifts collect the necessary information to verify the object hypothesis; and
- tracking mode: smooth pursuit movement.

The third part of the gaze control is somewhat like an intelligent instance because it selects the model appropriate to the current situation. Nevertheless, the selection of the right model is also automatic. Therefore, we call this control level behavior. Figure 27.6 shows the attention model that is inherent in NAVIS and described in more detail in the following sections.

27.5.2 Attractivity representation

NAVIS selects an attention window around the current fixation point for further processing. This window is analyzed with regard to some primitive features. These features could be either static or dynamic. The currently implemented static features are oriented edge elements, homogeneous area elements, and color segments. These features are detected in parallel, categorized, and sorted in several feature maps. At present, there are 12 maps for the oriented edge elements (angle resolution 15°), 13 maps for the oriented area elements (angle resolution 15° plus one map for the segments with an eccentricity near zero), and 6 color maps (60° intervals on a color circle).

The feature maps are analyzed with regard to some obvious conspicuities. Roughly summarized this analysis contains the evaluation of geometric dimensions such as the center of gravity, symmetry, orientation, and eccentricity and a measure for the local color contrast.

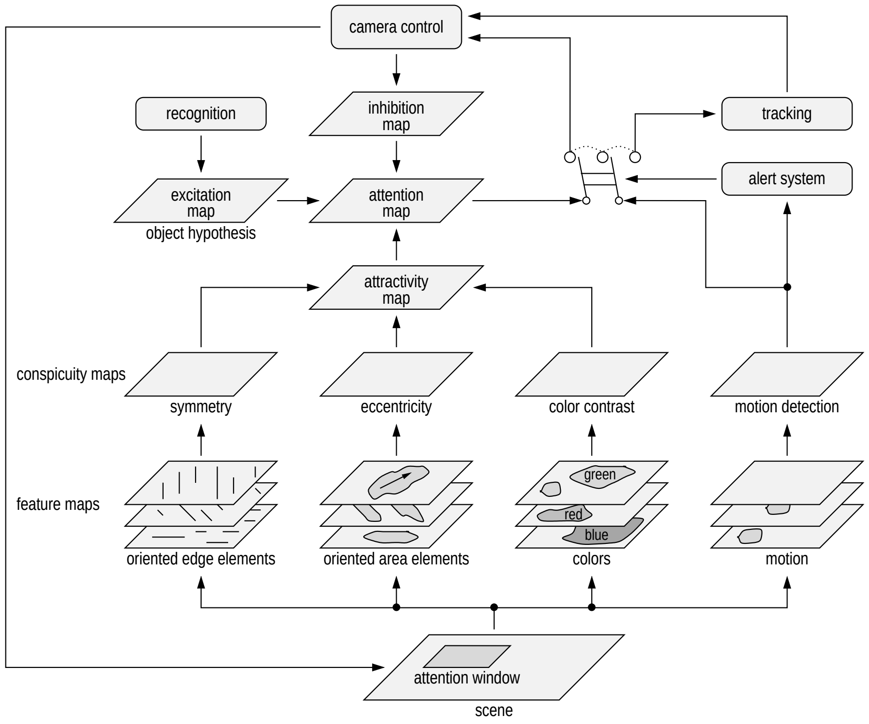


***Figure 27.6:** Attention model.*

The result of the conspicuity analysis is one map per feature containing the salient points. We call these salient points *attractivity points*. They are potential candidates for the next point of attention (POA) that holds the coordinates for the following camera shift. The attractivity points of the different conspicuity maps are sorted taking into account their saliency and then they are integrated into the attractivity map.

Symmetry. Artificial objects are often characterized by their salient symmetry. Hence, a symmetry analysis is also part of the attention model in NAVIS. The feature maps of the oriented edge elements (see Fig. 27.6) contain the information relevant to the generation of the conspicuity map symmetry. The oriented edge images are the basis for a multitude of NAVIS algorithms and therefore are described in detail in Section 27.3.1. Due to run-time considerations only one scale space with an angle resolution of 15° (resulting in 12 Gabor filter outputs) is evaluated in the attractivity representation. For each point in the feature maps the activity is summed up in a small range around a fixed radius perpendicular to the orientation of the edges. Subsequently, the results of all 12 feature maps are summed up and stored in so-called radius images. This procedure is done for several radii with adjacent summation ranges. A maximum operation on all radius images determines the attractivity points.

Eccentricity. The feature maps of the oriented area elements are based on a gray-level segmentation (see Section 27.3.2). Because the generation of the feature maps is solely scene data-driven, the segmentation process (region growing plus subsequent merging) can only take into account such common criteria as variance of the gray values, contrast to the surrounding, etc. This branch of the attractivity representation supplies attractivity points that cannot be detected by the symmetry analysis: objects without a conspicuous symmetry or objects partly occluded. The resulting segments are sorted concerning their orientation by a principal component analysis. The coordinates of the attractivity points are determined by calculating the center of gravity for each segment. The magnitude of the conspicuity of the attractivity points depends on the eccentricity of the segments. Circular objects (often detected by the symmetry operator) are related to low attractivities while line-shaped objects are related to high attractivities.

Color contrast. The branches of the attractivity representation described so far are based on a gray-level analysis. Specifically, they cannot cope with isoluminant object borders. Therefore, the evaluation of color is desirable. The color feature maps are categorical in nature. The category membership is determined in the HVC color space and depends only on the hue angle (see Section 27.3.3). Typically we use six categories: red; orange; yellow; green; blue; and purple. The corresponding conspicuity map assesses the color contrast of previously segmented regions to their surroundings. The segmentation is performed in the MTM color space and is based on a region-growing method (centroid linkage). The centers of gravity of conspicuous segments mark the attractivity points. The magnitude of an attractivity point is the mean gradient along its border with the neighbor segments. Here, the gradient is defined as the Euclidean distance between the mean colors of two segments.

27.5.3 Gaze control

The next POA is not determined solely by the activities in the attractivity map $\mathcal{AM}$ but also by the current model given by the behavior. In the hypothesis generation mode the inhibition map $\mathcal{IM}$ is the only additional map to be considered. It prevents the gaze control from paying attention to already fixated points in subsequent frames by reducing their attractivity exponential. Thus, the influence of already fixated points grows again with time. Not only the current POA but also the attractivity points in its surrounding are inhibited depending on an exponential distance function. The POA functions as the reference point to form an object hypothesis. In the hypothesis validation mode one more map has to be considered. The excitation map $\mathcal{EM}$ now contains those points $p_{\mathcal{EM}}$ belonging to a generated object hypothesis. These are points stored for a learned object and necessary to recognize it again. The excitation map makes it more difficult for the recognition module to switch between object parts (so-called forms) belonging to different hypotheses. Equation (27.8) shows the calculation of the candidates for the next POA. These candidates are stored in the attention map $\mathcal{POAM}$. Details of the recognition process are described in Section 27.6.

$$
\begin{aligned}
v(p_{\mathcal{POAM}}) = v(p_{\mathcal{AM}}) & \prod_{p_{\mathcal{IM}} \in \mathcal{IM}} (1 - v(p_{\mathcal{IM}}) \cdot d_1(p_{\mathcal{AM}}, p_{\mathcal{IM}})) \\
& \prod_{p_{\mathcal{EM}} \in \mathcal{EM}} (1 + v(p_{\mathcal{EM}}) \cdot d_2(p_{\mathcal{AM}}, p_{\mathcal{EM}}))
\end{aligned}
\tag{27.8}
$$

Here, $v(x)$ denotes the priority of a point x in $[0,1]$, $d(x,y)$ denotes a distance measure for x and y in $[0,1]$ (1 for identical locations).

The behavior control exchanges the hypothesis generation or validation model for the tracking model whenever the motion detector in NAVIS records motion, that is, the motion detector is always running between two gaze shifts (camera movements). We choose this behavior for NAVIS because a moving object might impair the current hypothesis generation or validation mode by adding new features to the scene or covering some relevant features of the static objects. At present, the behavior control of NAVIS can only cope with one moving object at any time. A more sophisticated behavior would be to track the nearest or fastest object if there were more than one. The tracking module of NAVIS is described in Section 27.7.

Figure 27.7 shows a short saccade of the gaze control. In this example the camera movement is solely scene data-driven. Other experiments are described in [11] and [12]. Experiments with model-driven gaze control are also introduced in the following sections.

27.6 Object recognition

Object-recognition modules are essential for autonomous vision systems. One can distinguish between 2-D and 3-D object recognition according to the available information about the model objects (see also Section 1.2). Within NAVIS there are two independent recognition systems. Both work with real-world objects (toy cars).

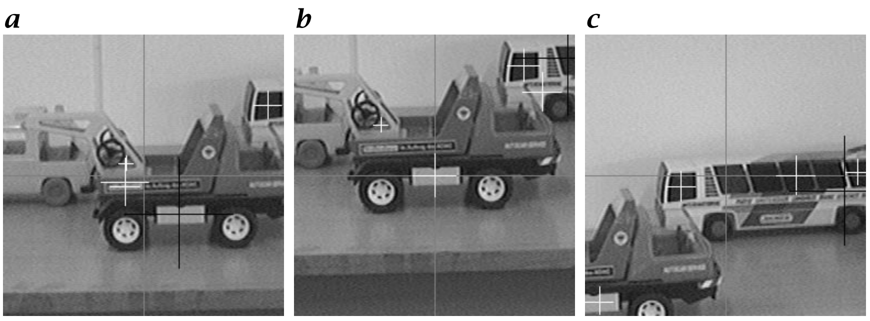


Figure 27.7: *Saccade: the current POA is marked by the gray cross, the next POA is marked by the black cross, and the attractivity points with lower priority are marked by the white crosses.*

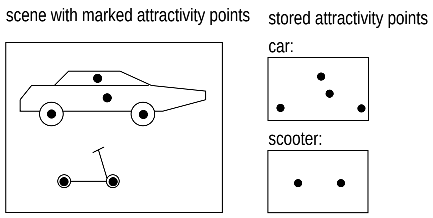

scene with marked attractivity points
stored attractivity points
car:
scooter:

Figure 27.8: *Principle of the hypothesis generation.*

27.6.1 Two-dimensional object recognition

The 2-D object-recognition system used within NAVIS is based on results in physiological and psychological research on the visual system of primates. The basic principle of the recognition system is the quite constant formation of attractivity points. An attractivity point is seen as a point with special properties according to symmetries and area features. A survey over appropriate algorithms and examples of images with marked attractivity points are presented in Section 27.5. The major assumption to form an object hypothesis is the recognition of a similar configuration of attractivity points in the scene. This principle is outlined in Fig. 27.8. A hypothesis mainly provides information about the location of an object and a first guess on the objects identity (*Where* processing path). After establishing a hypothesis the systems looks for *a priori* specified object parts (further called forms). By matching these forms the hypothesis is accepted or rejected, respectively (*What* processing path). The division into a where and what part is motivated by biological findings. Accordingly, the *infero temporal cortex* is concerned with discriminating objects. On the other hand, the *posterior parietal cortex* is responsible for establishing spatial relations.

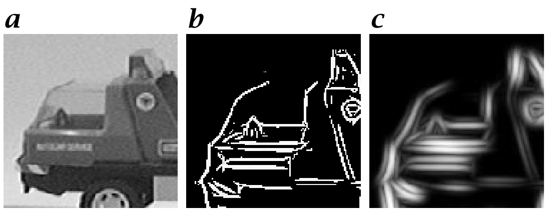


Figure 27.9: ***a*** *Exemplary object;* ***b*** *learned form;* ***c*** *presented form for matching.*

The basic system units are displayed in Fig. 27.1. The camera unit controls pan/tilt movements based on location provided by the control unit. Within the feature unit interesting points due to symmetry and area features are computed. One attention point is selected among all attractivity points in the attention unit. These three units were described in previous sections. During the hypothesis-validation mode subimages with dilated edges are generated. By matching these edges with skeletonized edge images a certain amount of object distortion is tolerated. Figure 27.9 shows such edge images.

Recognition process. The object-recognition module within the control unit is responsible for building hypotheses and for validating them. In Fig. 27.10 the major steps during the recognition process are depicted. In an off-line learning stage the objects together with marked forms are presented. Generated attractivity points and the spatial relations of the forms are learned. In the hypothesis-generation mode generated attractivity points are tested for stored object configurations. In this way the attention point serves as the reference point. The object with the highest match is taken as the next hypothesis. During the validation of a hypothesis the unit tries to match the stored forms against the presented forms. The relative position of the next form is given to the camera unit for consecutive foveation. The learned attractivity points of the hypothesized object are put into the excitation map to ensure that the same object is given attention in the next cycle. On the other hand, examined points are marked in the inhibition map to avoid further influence.

Experiments. By using active vision capabilities the system is able to explore an environment autonomously. Thus it can accordingly respond to changing scenes and new appearing objects. In Fig. 27.11 an exemplary recognition process is outlined. Step by step all object forms are examined and matched.

Recognition problems due to object transformations have to be overcome in further investigations. Here, the rigid representation of at-

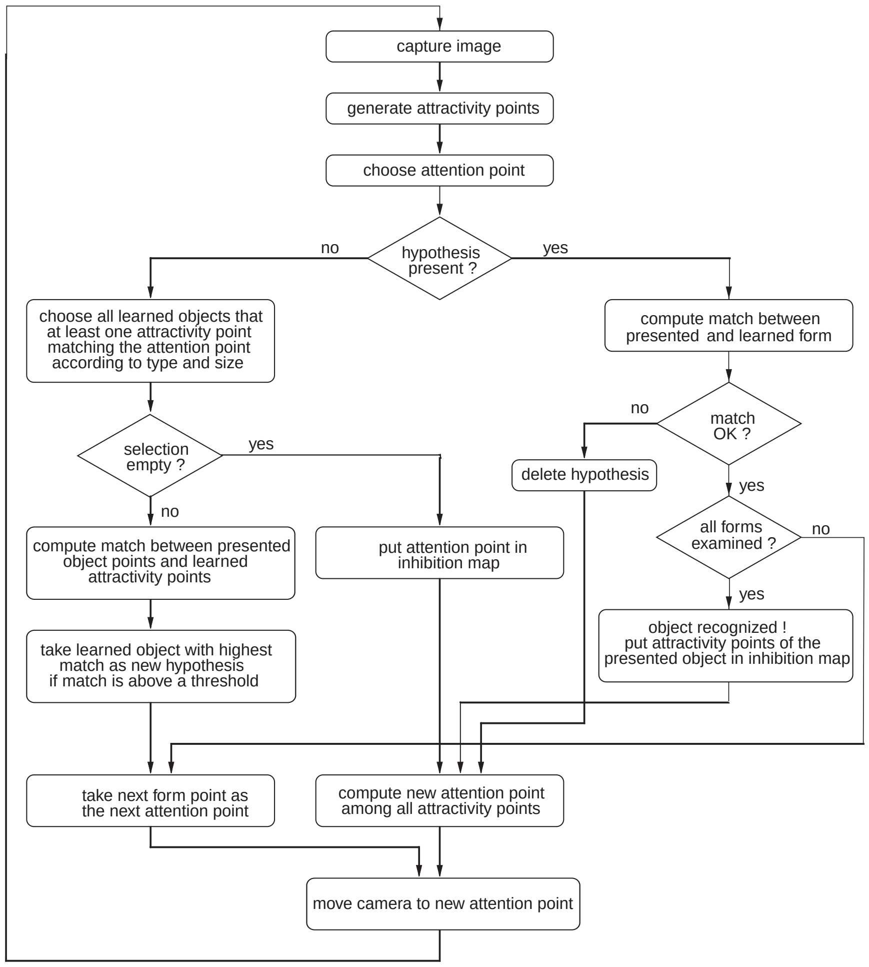

capture image
generate attractivity points
choose attention point
hypothesis present ?
no
yes
choose all learned objects that at least one attractivity point matching the attention point according to type and size
compute match between presented and learned form
match OK ?
no
delete hypothesis
yes
selection empty ?
yes
no
all forms examined ?
no
yes
compute match between presented object points and learned attractivity points
put attention point in inhibition map
object recognized ! put attractivity points of the presented object in inhibition map
take learned object with highest match as new hypothesis if match is above a threshold
take next form point as the next attention point
compute new attention point among all attractivity points
move camera to new attention point

Figure 27.10: *Major steps during recognition.*

tractivity points and object forms have a negative influence. A more detailed description of the system can be found in Götze et al. [13].

27.6.2 Three-dimensional object recognition

Three-dimensional object-recognition capabilities for real-world scenes are a major step towards a fully autonomous active vision system. As a result of psychophysic research a viewer-centered approach was chosen. An autonomous and effective method to form and administer view classes had to be found. Similarly, in the view sphere neighboring views should be classified in one view class. Possibly nonunique views should not disturb the recognition process over a longer period. By analyz-

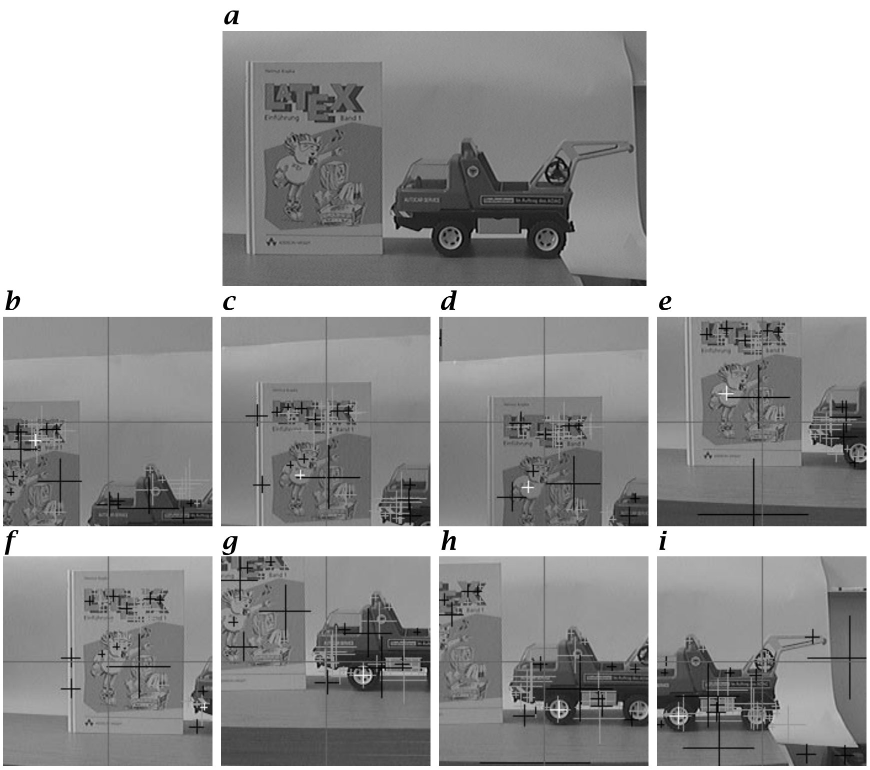


Figure 27.11: *Exemplary recognition process. The generated attractivity points are marked black and gray, the attention point is marked white:* ***a*** *test environment;* ***b*** *,* ***c*** *the system is looking for learned attractivity point configurations;* ***d*** *,* ***e*** *after building up the hypothesis "book" it searches for the specified object forms;* ***f*** *,* ***g*** *looking for a new hypothesis;* ***h,*** ***i*** *validating hypothesis "lorry."*

ing the temporal context a correct sequence classification is ensured despite such troubling views. In Fig. 27.12a the representation hierarchy used is depicted. Views are first classified to view transitions (sequences). The sequences are associated with objects in the following.

System architecture. In Fig. 27.12b the recognition system is shown. At present, the system is not fully integrated in NAVIS (cf. Fig. 27.1), and attention mechanisms are not used yet. The matching of views to view classes by an adapted *Dynamic Link Matching* build the core of the system. In the following the processing steps are outlined:

- *feature extraction:* Directed and oriented edges are used as the feature base by applying two Gabor filter sets with different mean frequencies (see Section 27.3.1). Twelve orientations per filter set are

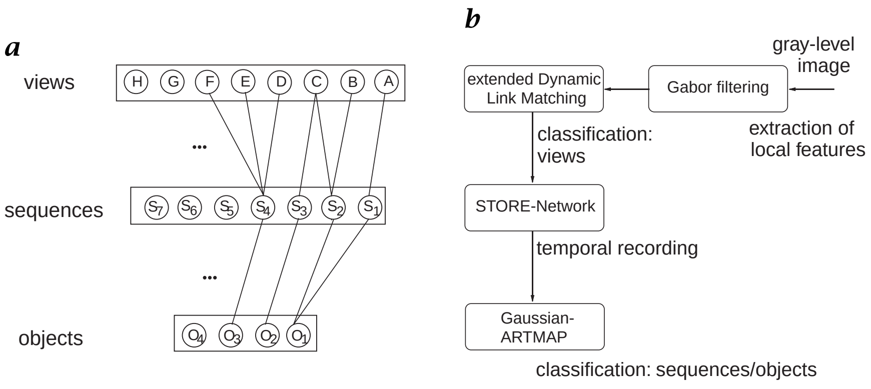


Figure 27.12: ***a*** *Representation hierarchy;* ***b*** *block diagram of the recognition system depicting the relations of the processing steps to the implemented algorithms.*

distinguished resulting in a 24-D feature space. To reduce computational costs the 256×256 pixel feature images are subsampled to 64×64 pixel arrays;

- *view classification:* To classify the presented views against the learned view classes a modified version of the *dynamic link matching* [14, 15] was used. The assignment of a local transformation of a feature point to its local neighborhood provides the basic idea. Two layers exist in which the presented (cells a) and learned patterns (cells b) are stored. Between these layers interlayer-connections (called *dynamic links*) are inserted in order to model a short-term memory. Thus stable connections are established between similar patterns during the iteration process. Scene regions (so-called blobs) are selected at random and matching regions in the model layers are activated. A matching score using the relative frequency of a simultaneous activation of two cells a and b, the coactivation C_{ba}, was developed. The model cells b are interpreted as the nodes of a grid. The connections to neighboring neurons form the grid edges. The node for a cell b is located at the center of gravity of its C_{ba} values with subsequent smoothing of the grid over neighboring nodes. In Fig. 27.14 two resulting grids are depicted. The matching score is derived from the degree of grid distortion. Two geometrical methods were implemented: one using the variance of the node distances, and one analyzing the orthogonality of the grid. Both showed similar behavior. There are also local excitatoric and global inhibitoric connections within the feature layers. The original dynamic link algorithm had to be modified at several steps due to the inherent similarity of all views within a sequence (see Fig. 27.13 for an exemplary sequence);

Figure 27.13: *Part of an exemplary object sequence.*

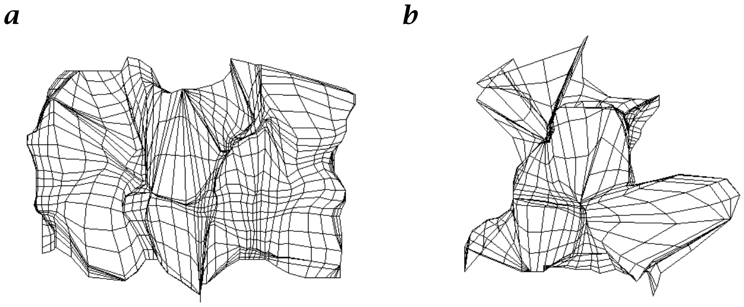


Figure 27.14: *Coactivation grids derived from Dynamic Link Matching:* ***a*** *almost uniform grid; presented object could be matched;* ***b*** *distorted grid; presented object could not be matched.*

- *recording of temporal context:* Through the temporal recording not only the appearance of events but also the temporal relations to each other should be encoded. Such an *item and order* encoding is provided by the *STORE* architecture according to [16]. The network stores past events and after a specific delay new events can exert influence. One can directly feed the output to a long-term memory;
- *object classification:* For the classification of object sequences a *Gaussian ARTMAP* network according to [17] is used. The network is constituted of two *ART* subnetworks ART_a, ART_b and a map field. The sequence vectors delivered by the STORE network are fed into the ART_a network. Within the networks object classes are represented by a Gaussian distribution. Every class i is described by M-dimensional vectors, whereby the mean values μ_i and the standard deviation σ_i together with an occurrence counter n_i are stored. After presenting the sequence vector to the network the classification result can be read from the map field.

Experiments. The system was trained with three objects (similar toy cars). For each object 25 single views were generated by rotating them in 15° steps. At the end of the training phase the 75 training views were classified in 55 almost evenly spread view classes. Figure 27.15 depicts

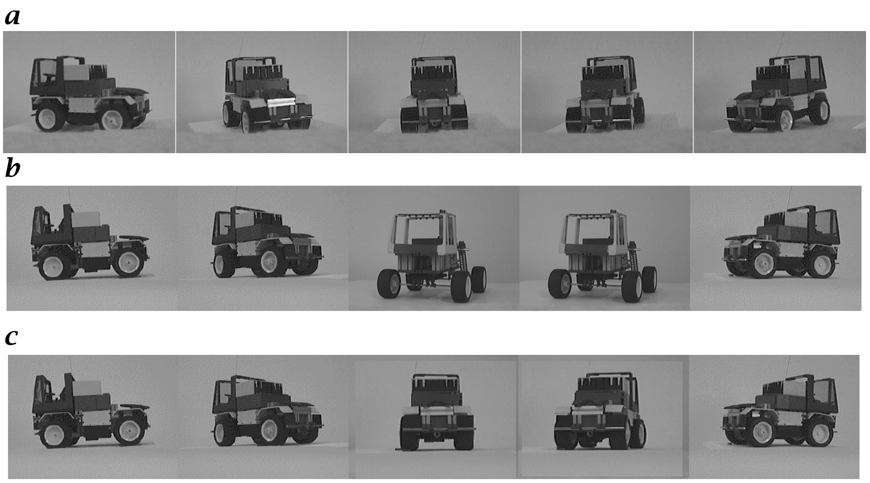


Figure 27.15: *Classification of an object sequence:* ***a*** *presented sequence;* ***b*** *assigned view classes (note the two misclassifications at views 3 and 4);* ***c*** *resulting correct object classes after applying the temporal recordings.*

a presented, not learned view sequence together with the resulting view classes. Here, the advantage of using the temporal context of object appearance is illustrated. Despite a misclassification the resulting object assignment is always correct.

By incorporating several approaches known from the literature a complete 3-D object recognition system was constructed. Its special strength results from the integration of temporal relations into the object representation. Further effort is necessary to use active vision capabilities, for example, attention mechanisms (see Section 27.5) and figure-ground separation (see Section 27.4). Thus an autonomous registration of object sequences will be possible. A tight coupling with tracking mechanisms (see Section 27.7) seems promising. More details of the approach presented here can be found in Massad et al. [18].

27.7 Motion estimation and object tracking

The estimation of object motion from a sequence of images provides essential information for an observer interacting with its environment. The apparent motion of patterns in the image plane may result from motion of objects in the scene, observer motion, or both. It can also be influenced by the modification of lighting conditions, which, however, should not further be considered here.

27.7.1 Displacement vector estimation

Methods for the estimation of velocity vector fields can be classified according to the size of the region from which input data is obtained. Following this scheme, image-difference techniques depending on pixel-sized regions have the smallest region of input. Methods using spatiotemporal derivatives of an image sequence produce dense flow fields that have the property of being less local, but certain assumptions on spatiotemporal image changes must be made in order to find a unique solution for the *optical flow field* [19, 20, 21].

A third category of feature-based techniques utilizes more complex and therefore better distinguishable image features. They allow the detection of larger object displacements as certain limitations concerning the aperture problem can be eliminated. Structures in the image are represented in an appropriate manner to obtain a unique parametrization of object features. Many approaches have been introduced so far that use different features to generate object representations. A popular class of features consists of the corners of objects [22, 23] or line segments [24]. Kories [25] and Koller [26] use a heuristically defined feature extractor, the *monotonicity operator*, which separates the image pixels into nine classes representing topological structures of the gray-value function.

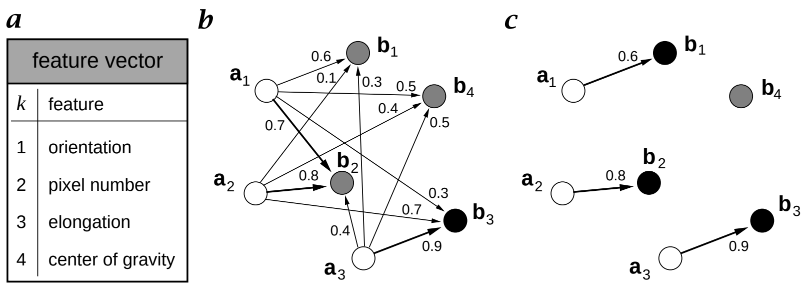
The approach presented here tries to merge simple features that are associated with the same spatially extended structure into a single parametrization describing this structure. A filtering process with a set of Gabor filters with differing orientations and scales results in a decomposition of the input image into a corresponding number of filter responses (see Section 27.3). Because the real and imaginary part form a quadrature filter pair the magnitude of the complex filter response is invariant against translation. It can be considered as a measure for the local energy of the input signal within a certain bandwidth and orientation.

A comprehensive description of the underlying concepts and the realization of the tracking module can be found in Springmann [27].

27.7.2 Correspondence analysis

The magnitude of the filter response serves as input for a dilatation process that tries to combine all points belonging to a certain spatial structure. First, a labeling process segments the spatial magnitude distribution into distinct clusters. For each cluster a set of parameters is calculated. In particular, these are

- orientation;
- number of points;



feature vector	
k	feature
1	orientation
2	pixel number
3	elongation
4	center of gravity

***Figure 27.16:** Computation of similarities: **a** components of a single feature vector; **b** and **c** example for correspondence evaluation between feature vectors for $m = 3$ and $n = 4$. Set $\mathcal{A}$ is represented by white circles; **b** state after the first iteration step. All connections are labeled with the accompanying similarity measure. Elements from $\mathcal{B}$ that have no or ambiguous connections are depicted as gray circles. Thick arrows denote preliminary established correspondences; **c** final result after applying the plausibility constraint. Feature vector $\boldsymbol{b}_4$ remains unconnected.*

- elongation; and
- coordinates of the center of gravity.

All parameters build up the components of a more complex feature vector attached to a certain cluster that serves as its identifier (Fig. 27.16a). Hence, a feature vector must be unique in order to facilitate discrimination between image structures. It could be shown in several experiments that the representation chosen in the foregoing meets these requirements.

After transferring all image structures into feature vectors, they have to be compared to those of a previously processed image by some measure of similarity. Consider two frames A and B each containing a number of n or m structure elements, respectively. The vectors $\boldsymbol{a}_i, i = 1, \ldots, m$ and $\boldsymbol{b}_j, j = 1, \ldots, n$ denote the feature vectors for structure element i or j, respectively. Then a similarity measure s_{ij} between two feature vectors $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$ should only be computed if both orientations are identical. For all other components of the feature vector the similarity measure is computed as follows:

$$s_{ij} = \begin{cases} \prod_{k=2}^{4} \frac{\min(a_{ik}, b_{jk})}{\max(a_{ik}, b_{jk})}, & a_{i1} = b_{j1} \\ 0, & \text{otherwise} \end{cases} \tag{27.9}$$

where a_{ik} and b_{jk} denote the kth components of the feature vectors $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$ that form the two sets $\mathcal{A}$ and $\mathcal{B}$ computed from two subsequent frames. If $a_{ik} = b_{jk}$, the corresponding kth factor of the product term in the foregoing is one, otherwise the value lies in the interval $[0, 1)$ depending on the ratio of both components. To determine unique correspondences between the elements of $\mathcal{A}$ and $\mathcal{B}$ an iterative algorithm is applied. First, each vector $\boldsymbol{a}_i$ from $\mathcal{A}$ is matched against all vectors $\boldsymbol{b}_j$ from $\mathcal{B}$ (Fig. 27.16b). Two vectors $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$ with maximal similarity measure over all $s_{ij}, j = 1, \ldots, m$ are marked as candidates for a pair of corresponding feature vectors. After this first iteration step all vectors from $\mathcal{A}$ are uniquely connected to a vector in $\mathcal{B}$, but a certain vector in $\mathcal{B}$ may have either none or multiple connections to elements in $\mathcal{A}$ (Fig. 27.16b). This is in contradiction to a plausibility constraint that states that a transformation from $\mathcal{A}$ into $\mathcal{B}$ has to be bijective (each element from $\mathcal{A}$ can have at least one counterpart in $\mathcal{B}$).

To avoid multiple linking only those feature vectors with the best matching result remain connected. As a consequence, several vectors from $\mathcal{A}$ are no longer connected to a vector in $\mathcal{B}$ and must be re-linked in an additional iteration step. The algorithm stops if no elements from $\mathcal{A}$ remain with multiple relations to $\mathcal{B}$. As shown in Fig. 27.16c, this scheme results in $n - m$ unconnected vectors in $\mathcal{B}$ (if $m < n$).

27.7.3 Tracking process

The unique correspondence of feature vectors can be expressed in terms of a motion vector field aligning structures in consecutive frames. The difference of the centers of gravity in association with the similarity of the feature vectors leads to a weighted motion field.

Figure-ground separation. The displacement of the moving object can be extracted from this vector field. Separation into regions characterized by uniform motion is performed by calculating a 2-D histogram over all motion vectors. A set of features describing uniform motion can be detected by searching the histogram for local maxima. One of these sets represents the background motion induced by the egomotion of the camera. Others correspond to moving objects characterized by oriented structures.

The background motion can be computed from the previous camera motion and the known inverse kinematics of the camera head. Hence, the group of motion vectors belonging to the background can be identified. If there is a moving object in the observed region, as assumed here, its motion is represented by a larger group of motion vectors different from the background motion. Because the motion vectors associated with a certain object and the appropriate centers of gravity in the corresponding feature vectors are known, the egomotion and position of a moving object in the image plane can be calculated.

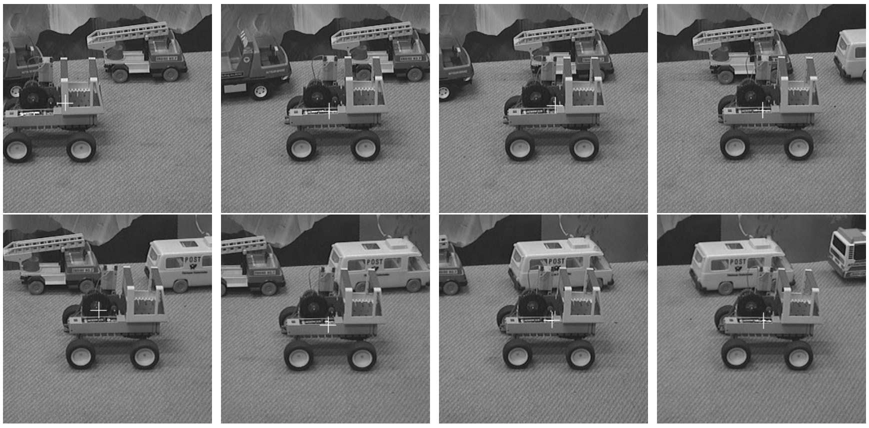


***Figure 27.17:** Example for a tracking sequence (from upper left to lower right): In the first frame the vehicle is detected as a moving object and is marked with a cross. After an initial gaze shift it is centered in the image plane.*

Object tracking. To center a selected object in the image plane the camera has to compensate the expected object motion. Hence, a motion prediction has to estimate the kinematic state of an object in the next frame by balancing the differences between the measured and estimated object state. This task is performed by a linear *Kalman filter*, as it represents an optimal filter minimizing the difference between the measured and predicted system state over time. Linear Kalman filters use an internal state representation of the moving object that is affected by noise. Therefore, this type of filters needs an explicit motion model that is assumed here to be a constantly accelerated motion.

If a moving object enters the observed region or the motion prediction has failed, a correction saccade has to be performed. As a result, features can be displaced up to half the size of a frame, many new features will appear, and others will vanish. Correspondence of these features cannot be established instantly. Therefore the next camera motion relies only upon the current state of the motion-prediction process.

Experiments. Figure 27.17 shows a typical image sequence from a tracking process. In the first frame the moving toy car appears at the left border of the image in the tracking initialization phase by taking two frames and computing the motion and center of gravity of the moving object. Then a gaze shift centers the object in the image plane. As the camera follows the object the point of fixation is kept on the center of the vehicle.

Our tracking system uses complex features derived from image structures to establish correspondences between subsequent frames. Experiments have shown that this representation provides a reliable estimation of displacement vectors related to moving objects. At present, more work has to be done on a more reliable and flexible motion segmentation in the case of several moving objects. In the future, depth cues from the stereo module will be used to distinguish between moving objects. In addition, the determination of a stable fixation point has to be improved.

27.8 Conclusion

The active vision system presented here is inspired by visual information processing as it can be observed in biological systems. NAVIS gathers information from the environment that is interpreted and leads to appropriate actions. These actions must be purposive in order to solve certain tasks. A special behavior-control entity analyzes the information provided by underlying processing units and is responsible for a proper task scheduling. It can receive interrupts in order to enable reactions on exceptional events. Once a task has been selected the data flow between subordinated units can be rearranged or additional units can be included into data processing.

The chapter described the current state of the system. At the moment, the attention unit and the 2-D object recognition are already incorporated in an integrated system. The 3-D object recognition and the tracking modules are momentarily working on a stand-alone basis and will be added to the system in the near future. Further work will concentrate on the implementation of more complex behaviors.

Acknowledgments

This project is partly funded by the Deutsche Forschungsgemeinschaft (DFG).

27.9 References

[1] Granlund, G. H., (1988). *Integrated analysis-response structures for robotics systems.* Technical Report, Computer Vision Laboratory, Linkoping University, Sweden. Report LiTH-ISY-I-0932.

[2] Drüe, S. and Trapp, R., (1996). Ein flexibles binokulares Sehsystem: Konstruktion und Kalibrierung. In *Aktives Sehen in technischen und biologischen Systemen*, B. Mertsching, ed., number 4 in Proceedings in Artificial Intelligence, pp. 32–39. St. Augustin: Infix.

[3] Guzzoni, D., Cheyer, A., Julia, L., and Konolige, K., (1997). Many robots make short work. *AI Magazine*, **18**(1):55–64.

[4] Jones, J. P. and Palmer, L. A., (1987). An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. *Jour. Neurophysiology*, **58**:1233–1258.

[5] Theimer, W. and Mallot, H., (1994). Phase-based binocular vergence control and depth reconstruction using active vision. *CVGIP: Image Understanding*, **60**(3):343–358.

[6] Trapp, R., (1996). *Entwurf einer Filterbank auf der Basis neurophysiologischer Erkenntnisse zur orientierungs- und frequenzselektiven Dekomposition von Bilddaten*. Technical Report NAVIS 01/96, Universität-GH-Paderborn.

[7] Jähne, B., (1997). *Digital Image Processing - Concepts, Algorithms, and Scientific Applications*. Berlin: Springer.

[8] Miyahara, M. and Yoshida, Y., (1988). Mathematical transform of (R,G,B) color data to Munsell (H,V,C) color data. *Visual Communication and Image Processing*, **1,001**:650–65.

[9] Trapp, R., Drüe, S., and Mertsching, B., (1995). Korrespondenz in der Stereoskopie bei räumlich verteilten Merkmalsrepräsentationen im Neuronalen-Active-Vision-System NAVIS. In *Mustererkennung 1995*, G. Sagerer, S. Posch, and F. Kummert, eds., pp. 494–499. Berlin: Springer.

[10] Lieder, T., Mertsching, B., and Schmalz, S., (1998). Using depth information for invariant object recognition. In *Dynamische Perzeption*, S. Posch and H. Ritter, eds., pp. 9–16. St. Augustin: Infix.

[11] Bollmann, M., Hoischen, R., and Mertsching, B., (1997). Integration of static and dynamic scene features guiding visual attention. In *Mustererkennung 1997. Informatik aktuell*, E. Paulus and F. M. Wahl, eds., pp. 483–490. Berlin: Springer.

[12] Mertsching, B. and Bollmann, M., (1997). Visual attention and gaze control for an active vision system. In *Progress in Connectionist-Based Information Systems. Volume 1*, N. Kasabov, R. Kozma, K. Ko, R. O'Shea, G. Coghill, and T. Gedeon, eds., pp. 76–79. Singapore: Springer.

[13] Götze, N., Mertsching, B., Schmalz, S., and Drüe, S., (1996). Multistage recognition of complex objects with the active vision system NAVIS. In *Aktives Sehen in technischen und biologischen Systemen*, B. Mertsching, ed., pp. 186–193. Sankt Augustin: Infix.

[14] Konen, W., Maurer, T., and von der Malsburg, C., (1994). A fast link matching algorithm for invariant pattern recognition. *Neural Networks*, **7**:1019–1030.

[15] von der Malsburg, C., (1981). *The correlation theory of brain function*. Technical Report, Max-Planck-Institut für Biophysikalische Chemie, Göttingen.

[16] Bradski, G., Carpenter, G., and Grossberg, S., (1992). Working memory networks for learning temporal order with application to three-dimensional visual object recognition. *Neural Computation*, **4**:270–286.

[17] Williamson, J., (1996). Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps. *Neural Networks*, **9** (5):881–897.

[18] Massad, A., Mertsching, B., and Schmalz, S., (1998). Combining multiple views and temporal associations for 3-D object recognition. In *Proceedings of the ECCV '98*. Freiburg.

[19] Barron, J., Fleet, D., and Beauchemin, S., (1994). Performance of optical flow techniques. *International Jour. Computer Vision*, **12**(1):43–77.

[20] Horn, B. and Schunk, B., (1981). Determining optical flow. *Artificial Intelligence*, **17**:185–203.

[21] Nagel, H.-H., (1987). On the estimation of optical flow: Relations between different approaches and some new results. *Artificial Intelligence*, **33**(3): 299–324.

[22] Reid, I. and Murray, D., (1996). Active tracking of foveated feature clusters using affine structure. *International Journal of Computer Vision*, **18**(1):41–60.

[23] Smith, S. M., (1995). ASSET-2: Real-time motion segmentation and object tracking. In *Proc. of the Fifth Int. Conf. on Computer Vision*, pp. 237–244.

[24] Andersson, M., (1994). *Tracking Methods in Computer Vision*. PhD thesis, Computational Vision and Active Perception Laboratory, Royal Institute of Technology, Stockholm.

[25] Kories, R. R., (1985). *Maschinelles Bewegungssehen in natürlichen Szenen: Die Auswertung von Bildfolgen gestützt auf Bildmerkmale*. Dissertation, Fachbereich Biologie, Johannes Gutenberg Universität, Mainz.

[26] Koller, D., (1992). *Detektion, Verfolgung und Klassifikation bewegter Objekte in monokularen Bildfolgen am Beispiel von Straßenverkehrsszenen*. Dissertation, Infix, Sankt Augustin.

[27] Springmann, S., (1998). *Merkmalskorrespondenz in realen Bildsequenzen zur Bewegungsschätzung und Verfolgung von Objekten*. Master's thesis, Fachbereich Informatik, Universität Hamburg.

28 Dynamic Vision for Perception and Control of Motion

Ernst Dieter Dickmanns[1], and Hans-Joachim Wünsche[2]

[1]Universität der Bundeswehr München, Neubiberg, Germany
[2]Schroff UK Ltd, Hemel Hempstead, Great Britain

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

28.1 Introduction

Digital image processing started almost four decades ago with rectification and preprocessing of images from remote sensing and from medical applications for human interpretation. Processing time usually was in the 1-h time range per image. For a long time, image processing was confined to snapshot images because the data volume of 10^4 to 10^6 pixel per image and the relatively low processing speed of computers up to the 1980s did not allow digital real-time video data processing.

Change detection in images attracted interest as data on the same scene but temporally well apart became available; this was of interest both for medical diagnosis and for military intelligence. Starting from this background, the idea of motion recognition and vehicle control by computer vision came about in the context of planetary rover vehicles [1]. Initial test vehicles had only the camera on board, but the computers required remained in the laboratory; they typically needed a quarter of an hour for image analysis. This was the state in the late 1970s, early 1980s [2]. The vehicles moved a short distance and had to wait, standing still as long as the image was analyzed. Each image was interpreted from scratch.

At that time, Dickmanns, coming from the field of control engineering, had the idea of developing machine vision from a different paradigm. Instead of choosing a visually complex environment and working with very low image frequencies (10^{-1} to 10^{-3} Hz) as usual at that time in "artificial intelligence," he decided to keep image frequencies rather high (larger than 10 Hz) and visual complexity of the scene to be handled correspondingly low. This low cycle time of 100 or fewer milliseconds (ms) would allow exploiting temporal and visual continuity in the scene because only small changes had to be accommodated. The change of location of the camera observing the scene was to be modeled so that predictions would allow search space for the same features to be reduced drastically.

Road scenes, especially roads built for high-speed driving, seemed to be sufficiently simple visually for providing the first field of application. The surfaces are smooth and well kept, and the lanes are marked with bright colors for easy recognition by humans; in addition, in order to keep the effect of acceleration levels on human passengers at high speeds low, both horizontal and vertical curvatures are limited in magnitude. The model for this type of road can easily be explained mathematically. Because vehicles are massive bodies that do not jump around but move steadily, their motion can be described by differential equation constraints including the effects of control and perturbation inputs.

All of this has lead to a completely different approach to machine vision than those studied by researchers from the field of artificial intel-

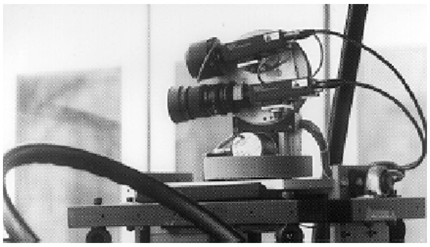


Figure 28.1: *Two-axis platform of Universität der Bundeswehr München (UBM) for viewing direction control from 1984.*

ligence. Recursive estimation based on incomplete measurement data, developed in the 1960s for trajectory determination in aerospace applications, was well established in the engineering disciplines at that time. These methods have been extended to image-sequence processing for full state reconstruction in 3-D space and time at the Universität der Bundeswehr München (UBM). The method obtained had been dubbed "4-D approach," in contrast to the 2-D, 2 1/2-D, and 3-D methods under discussion then, disregarding time as the fourth independent variable in the problem domain. The numerical efficiency and compactness in state representation of recursive estimation, which directly allowed control applications for generating behavioral capabilities, finally led to its widespread acceptance in the community dealing with autonomous vehicles (mobile robots) [3, 4, 5, 6, 7, 8].

28.1.1 Active vision

In the vision community, where 2-D image processing and the aspects of pure computational vision prevailed, many researchers remained skeptical of the recursive estimation approach. In the mid-1980s, it was realized by some researchers that accepting images for analysis just as they were obtained by chance, probably was not the most effective way to proceed. The "*active vision*" paradigm proposed by Bajcsy [9], Aloimonos [10], and Ballard [11] recommended adjusting image acquisition and parameter setting according to the task to be solved, especially, if image sequences were to be taken. A survey is given in Terzopoulos and Metaxas [12].

At UBM, a platform for active viewing direction control had been included from the very beginning when designing the test vehicle VaMoRs in 1984/1985. Because no suitable system could be found on the market, the first custom-made system of UBM was developed (see Fig. 28.1). The first applications were to steer gaze direction according to road curvature in order to keep the road at the desired lookahead range centered in the tele-image.

Next, in the early 1990s inertial rate sensors were added on the platform. Feeding the negative angular rate signal to the motor allowed stabilizing the viewing direction inertially despite the pitching moments during braking and acceleration maneuvers. Visual fixation of laterally positioned traffic signs with the telelens while approaching them was the application to follow [13]. The sign was picked up and tracked initially by a wide angle lens of a bifocal vision system; then a saccade of about 20° and a total duration of half a second was performed to center the sign for a moment in the tele-image only for a stabilized snapshot. This image was then passed on for analysis to a special processor, and the viewing direction was returned to the road section of interest.

In aeronautical applications such as on-board autonomous landing approaches, the viewing direction was controlled such that the landing strip was kept centered in the tele-image; angular perturbations from wind gusts were compensated for by negative inertial rate feedback. This direct interaction with inertial sensors turned out to be very beneficial and has led to what has been dubbed "dynamic vision."

28.1.2 Dynamic vision

Dynamic vision is the most general form of vision in scenes with several moving objects including the body carrying the camera set. There may or may not be a stationary background, but there are other moving objects in addition to the ego-vehicle. The motion of all of these objects may be partly intended, but partly due to unavoidable perturbations, such as pitching and rolling (banking) motion components for cars from an uneven surface, for ships from waves, or for aircraft from wind gusts. Under these conditions, joint inertial and visual sensing has several advantages:

1. The *inertial sensor* signals have almost no time delay as compared to several hundred milliseconds for image evaluation.
2. The sampling frequency is in the kilohertz range today, much higher than the 25-(or 30-) Hz standard video rate; this allows much faster response to perturbations.
3. The inertially sensed data correspond to accelerations (in the translational degrees of freedom (DOF)) and to angular rates (in the rotational DOF); these signals have a lead time relative to positions and angular orientations that are the second and first integrals of the former ones.
4. If there is no stationary background object in the scene, it is impossible to make a distinction between ego-motion and object motion only by vision. Inertial sensing is affected by *ego-motion* only; therefore, joint inertial and visual sensing provides a qualitatively different perception capability for active subjects.

5. Under strong ego-motion, especially with large angular rates, motion blur deteriorates the quality of images; inertial stabilization alleviates the vision task by taking this ego-motion component out of the image data stream, at least for certain periods when the eye can move freely.

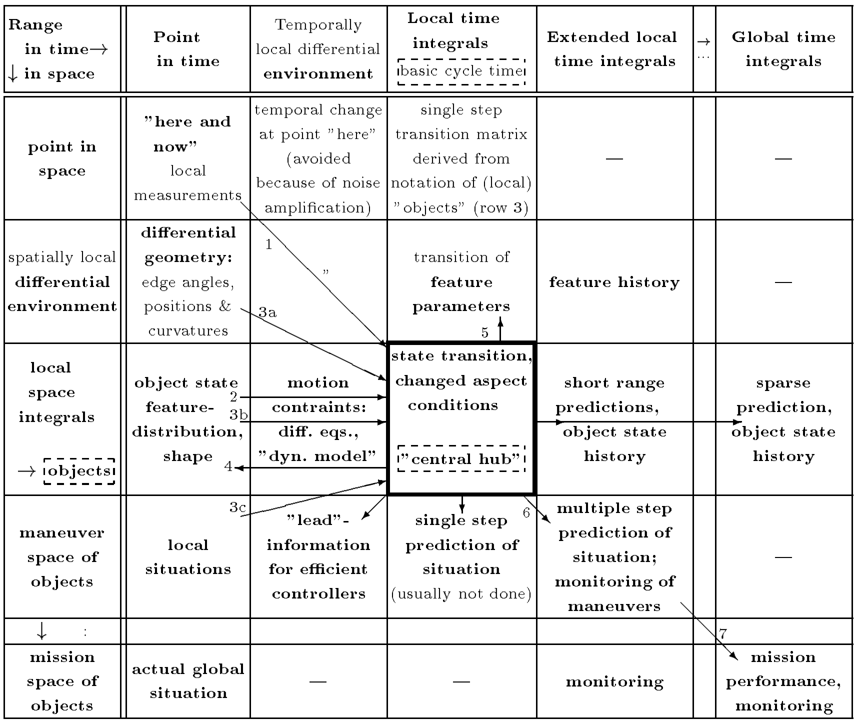
In order to make sense of jointly acquired inertial and visual data with different sampling rates and time delays as well as on different temporal integration levels, good *temporal models* have to be available in the interpretation process. It is exactly this information that is provided by the dynamic models underlying recursive estimation. Note that the same models are used in simulation for reproducing or forecasting system responses to control and to perturbation inputs. In control engineering, this knowledge is exploited to design feedback loops with prescribed response characteristics.

These temporal models of motion processes allow perceiving an extended presence, not just the state of objects at the point "now" on the time scale; this is of great importance for understanding the temporal aspects of a scene changing dynamically. Once this temporal dimension is represented in the perception process, its scale may be adjusted in order to be able to deal with processes of different extensions. Typically, for human perception and actions there are the notions of different time scales: 1) momentary events (duration: "an eye blink"); 2) maneuver elements (in the order of a second); 3) maneuvers (consisting of several maneuver elements) that perform a certain state change by typical control time histories; 4) mission elements; and 5) missions as one coherent chain of activities for achieving some goal. In addition, there are the natural time scales of days, years, and a life span, etc.

Thus, dynamic vision views the actual situation in a larger temporal context; various (heterogeneous) sensor signals are interpreted jointly exploiting the notion of time integrals and of terms defined in the time domain such as frequencies and time constants for typical linear systems dynamics. Delay times and precise spacing of events on the time scale are equally important for dynamic systems and their control.

28.1.3 Simultaneous representations on differential and multiple integral scales

For complex dynamical scenarios, inertial sensing in addition to vision is of great help; negative angular rate feedback to a viewing direction control device allows stabilizing the appearance of stationary objects in the image sequence. Measured accelerations and velocities will, via temporal signal integration, yield predictions for translational and rotational positions affecting the perspective mapping process. These predictions are good in the short run, but may drift slowly in the long run, especially when inexpensive inertial sensors are used.



Range in time→ ↓ in space	Point in time	Temporally local differential environment	Local time integrals basic cycle time	Extended local time integrals	→ …	Global time integrals
point in space	"here and now" local measurements	temporal change at point "here" (avoided because of noise amplification)	single step transition matrix derived from notation of (local) "objects" (row 3)	—		—
spatially local differential environment	differential geometry: edge angles, positions & curvatures	1 " 3a	transition of feature parameters 5	feature history		—
local space integrals → objects	object state feature-distribution, shape 2 3b 4	motion contraints: diff. eqs., "dyn. model"	state transition, changed aspect conditions "central hub"	short range predictions, object state history		sparse prediction, object state history
maneuver space of objects	local situations 3c	"lead"-information for efficient controllers	single step prediction of situation (usually not done) 6	multiple step prediction of situation; monitoring of maneuvers		—
↓ :						7
mission space of objects	actual global situation	—	—	monitoring		mission performance, monitoring

Figure 28.2: Differential and integral representations on different scales for dynamic perception.

These drifts, however, can easily be compensated by visual interpretation of static scene elements. The different time delays in the different signal pathways have to be taken into account in order to arrive at a consistent scene interpretation.

Combined use of inertial and visual sensing is well known from biological systems, for example, the vestibular apparatus and its interconnections to eyes in vertebrates. (Perturbations in this coordinated signal interpretation are conjectured to cause nausea in humans, both on boats and in motion simulators disregarding proper coordination.) In order to make optimal use of inertial and visual signals, both differential and integral representations on different scales in space and time are exploited simultaneously. Figure 28.2 shows the five categories introduced: local points and differentials as well as local, extended local, and global integrals. The upper left corner represents the point "here and now" in space and time where all interactions of a sensor or an actuator with the real world take place. Inertial sensors yield information on local accelerations (arrow 1 from field (1,1) to field (3,3) in the table) and turn rates of this point. Within a rigid structure of an object the turn rates are the same all over the body; therefore, the inertially measured rate signals (arrow 2 from field (3,1) to (3,3)) are drawn on the spatial object level (row 3).

The local surface of a structure may be described by the change of its tangent direction along some arc length; this is called curvature and is an element of local shape. It is a geometrical characterization of this part of the object in differential form; row 2 in Fig. 28.2 represents these local spatial differentials that may cause specific edge features (straight or curved ones) in the image under certain aspect conditions.

Single objects are considered to be local spatial integrals (represented in row 3 of Fig. 28.2), the shapes of which are determined by their spatial curvature distributions on the surface. In connection with the aspect conditions and the photometric properties of the surface they determine the feature distribution in the image. Because, in general, several objects may be viewed simultaneously, these arrangements of objects of relevance in a task context, called "geometrical elements of a situation," also are perceived and taken into account for behavior decision and reactive control. For this reason, the visual data input labeled by the index 3 at the corresponding arrows into the central interpretation process (field (3,3)) has three components: 3a) for measured features not yet associated with an object, the so-called detection component; 3b) the object-oriented tracking component with a strong predictive element for efficiency improvement; and 3c) the perception component for the environment that pre-shapes the maneuver space for the self and all the other objects. From this viewpoint, vision simultaneously provides geometrical information both on differential (row 2) and integral scales (rows: 3 for single objects, 4 for local maneuvering, and 5 for mission performance).

Temporal change is represented in column 2, which yields the corresponding time derivatives to the elements in the column to the left. Because of noise amplification associated with numerical differentiation of high-frequency signals ($\mathrm{d}/\mathrm{d}t(A\sin(\omega t)) = A\omega\cos(\omega t)$), this operation is usable only for smooth signals, such as for computing speed from odometry; especially, it is avoided deliberately to do optical flow computation at image points. Even on the feature level, the operation of integration with a smoothing effect, as used in recursive estimation, is preferred.

In the matrix field (3,2) of Fig. 28.2, the key knowledge elements and the corresponding tools for sampled data processing are indicated. Due to mass and limited energy availability, motion processes in the real world are constrained. Good models for unperturbed motion of objects belonging to specific classes are available in the natural and the engineering sciences, which represent the dependence of the temporal rate of change of the state variables on both the state and the control variables. These are the so-called "*dynamical models*." For constant control inputs over the integration period, these models can be integrated to yield difference equations, which link the states of objects in column 3 of Fig. 28.2 to those in column 1, thereby bridging the gap of column 2. In control engineering, methods and libraries with computer codes are available to handle all problems that arise. Once the states at one point in time are known these models deliver the corresponding time derivatives that are used instead of numerical derivatives of measured data.

Recursive estimation techniques developed since the 1960s exploit this knowledge by making state predictions over one cycle disregarding perturbations. Then, the measurement models are applied yielding predicted measurements. In the 4-D approach, these are communicated to the image-processing stage in order to improve image-evaluation efficiency (arrow 4 from field (3,3) to (3,1) in Fig. 28.2 on the object level, and arrow 5 from (3,3) to (2,3) on the feature extraction level). A comparison with the actually measured features then yields the prediction errors used for state update.

In order to better understand what is going to happen on a larger scale, these predictions may be repeated several to many times very rapidly in advance simulation assuming likely control inputs. For stereotypical maneuvers such as lane changes in road vehicle guidance, a finite sequence of "feedforward" control inputs is known to have a longer-term state-transition effect. These are represented in field (4,4) of Fig. 28.2 and by arrow 6; Section 28.4.8 will deal with these problems.

For the compensation of perturbation effects, direct state feedback well known from control engineering is used. With linear systems theory, eigenvalues and damping characteristics for state transition of the closed-loop system can be specified (field (3,4) and row 4 in Fig. 28.2). This is knowledge also linking differential representations to integral ones; low-frequency and high-frequency components may be handled separately in the time or in the frequency domain (Laplace transform) as usual in aerospace engineering. This is left open and indicated by the empty row and column in Fig. 28.2.

The various feedforward and feedback control laws that may be used in superimposed modes constitute *behavioral capabilities* of the autonomous vehicle. If a sufficiently rich set of these modes is available, and if the system is able to recognize situations when to activate these behavioral capabilities with parameters for achieving mission goals, the capability for autonomous performance of entire missions is given. This is represented by field (n,n) (lower right) and will be discussed in Sections 28.4.8–28.6. Essentially, mission performance requires proper sequencing of behavioral capabilities in the task context; with corresponding symbolic representations on the higher, more abstract system levels, an elegant symbiosis of control engineering and AI-methods can thus be realized.

28.2 Application areas discussed

Though the approach to dynamic vision is very general and has been adapted to other task domains also, only road and air-vehicle guidance will be discussed here.

28.2.1 Road vehicles

The most well-structured environments for autonomous vehicles are freeways with limited access (high-speed vehicles only) and strict regulations for construction parameters such as lane widths, maximum curvatures and slopes, on- and off-ramps, no same level crossings. For this reason, even though vehicles may be driven at high speeds, usually, *freeway driving* was selected as the first task domain for autonomous vehicle guidance by our group in 1985.

Six perceptual and behavioral capabilities are sufficient for navigation and mission performance on freeways: 1) *lane recognition* and lane following with adequate speed; 2) *obstacle recognition* and proper reaction, for example, transition into convoy driving or stopping; 3) recognition of neighboring lanes, their availability for lane change, and lane-changing performance; 4) reading and obeying *traffic signs*; 5) reading and interpreting navigation information, including proper lane selection; and 6) handling of entries and exits.

On well-kept freeways it is usually not necessary to check surface structure or to watch for humans or animals entering from the side. Nonetheless, safe reaction to unexpected events must be required for a mature autonomous system. On normal state roads the variability of road parameters and of traffic participants is much larger; especially, the same level crossings and oncoming traffic increase the relative speed between objects, thereby increasing hazard potential even though traveling speed may be limited to a much lower level. Bicyclists and pedestrians as well as many kinds of animals are normal traffic participants. In addition, lane width may be less on the average, and surface state may well be poor on lower-order roads, for example, potholes, especially in the transition zone to the shoulders of the roads.

In *urban traffic*, the situation may be even worse with respect to crowdedness and crossing of subjects. These latter-mentioned environments are considered to be not yet amenable to autonomous driving because of scene complexity and computing performance required.

However, driving on minor roadways with little traffic, even without macadam or concrete sealing, has been started for research purposes in the past, and may soon be performed safely with the increasing computing power currently becoming available. If it is known that the ground will support the vehicle, even *cross-country driving* can be done including obstacle avoidance. However, if compared to human capabilities in these situations, there is still a long way to go until autonomous systems can compete.

28.2.2 Air vehicles

As compared to ground vehicles with three degrees of freedom (DOF), full 6 DOF are available for trajectory shaping here. In addition, due to air turbulence and winds, the perturbation environment may be much more severe than on roads. For this reason, *inertial sensing* is considered mandatory in this task domain. In addition, visual navigation guidelines such as lanes on roads are not available once the aircraft is airborne at higher altitudes. Microwave and other electronic guidelines have been established instead.

Vision allows the pilot or an *autonomous aircraft* to navigate relative to certain landmarks; the most typical task is the landing approach to a prepared runway for fixed-wing aircraft, or to the small landing site usually marked by the capital letter H for a helicopter. These tasks have been selected for initial demonstrations of the capabilities of aircraft with the sense of vision. Contrary to other electronic landing aids such as *instrument landing systems* ILS or *microwave landing systems* MLS, machine vision also allows the detection of obstacles on the runway, the determination of motion, and how to react in a proper manner.

For flights close to the Earth surface, terrain formations may be recognized as well as buildings and power lines; thus, obstacle avoidance in nap-of-the-earth flights is a natural extension of this technique for unmanned air vehicles, both with fixed wings and for helicopters. For the latter, the capability of recognizing structures or objects on the ground and of hovering in a fixed position relative to these objects despite disturbances, will improve rescue capabilities and delivery performance. Motion control for fixed-wing aircraft and for helicopters differs greatly. However, by the use of proper dynamical models and control laws it has been shown that the 4-D approach allows each craft to be turned into an autonomous agent capable of fully automatic mission performance.

28.3 Sensory information used

As has been pointed out in Section 28.1.3, inertial information from an ego-body is crucial to understanding highly dynamic complex scenes in which it is one among several other moving objects. Without this information, in strongly perturbed environments situations sometimes could not be handled by vision alone.

The extremely high data rates of image sequences are both an advantage (with respect to versatility in acquiring new information on both

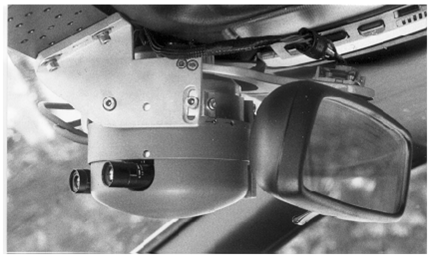


Figure 28.3: *Binocular camera arrangement of VaMP.*

environment and on other objects/subjects) and a disadvantage (with respect to computing power needed and delay time incurred until the information has been extracted from the data).

28.3.1 Conventional sensors

For ground vehicles, odometers, tachometers as well as sensors for positions and angles of subparts such as actuators and pointing devices are commonplace. For aircraft, pressure-measurement devices yield information on speed and altitude. Here, inertial sensors such as accelerometers, angular rate and vertical as well as directional gyros are standard. Evaluating this information in conjunction with vision considerably alleviates image-sequence processing. Based on the experience gained in air-vehicle applications, the inexpensive inertial sensors such as accelerometers and angular rate sensors, have also been adopted for road vehicles, because of their beneficial and complementary effects relative to vision. Part of this has already been discussed in Section 28.1.3 and will be detailed in the following.

28.3.2 Vision sensors

Because of the large viewing ranges required, a single camera as vision sensor is by no means sufficient for practical purposes. In the past, bifocal camera arrangements (see Fig. 28.3) with a wide angle (about 45°) and a telecamera (about 15° aperture) mounted relative to each other on a two-axis platform for viewing direction control have been used. In future systems, trinocular camera arrangements with a wide simultaneous field of view (> 100°) from two divergently mounted wide-angle cameras and a 3-chip color charge coupled device (CCD) camera will be used [14]. For high-speed driving on German Autobahnen, even a fourth camera with a relatively strong telelens will be added allowing lane recognition at a distance of several hundred meters. Cameras with large focal lengths will have to be stabilized inertially. All these data are evaluated 25 times/s, the standard European video rate.

28.3.3 Global positioning system (GPS) sensor

For landmark navigation in connection with maps a *global positioning system* (GPS) receiver has been integrated into the system in order to have sufficiently good initial conditions for landmark detection. Even though only the least accurate selective availability mode is being used, in connection with inertial sensing and map interpretation good accuracy can be achieved after some period of operation; GPS signals are available only once every second in the system used. Global position in geographic coordinates allows tapping into databases for *geographic information systems* (GIS) including visual landmarks.

28.4 Dynamic perception: The 4-D approach exploiting spatiotemporal models

Since the late 1970s, observer techniques as developed in systems dynamics [15] have been used at UBM in the field of motion control by computer vision [16, 17]. In the early 1980s, Wünsche did a thorough comparison between observer- and *Kalman filter* [18] realizations in recursive estimation applied to vision for the original task of balancing an inverted pendulum on an electrocart by computer vision. Since then, refined versions of the *extended Kalman filter* (EKF) with numerical stabilization (UDU^T-factorization, square root formulation) and sequential updates after each new measurement have been applied as standard methods to all dynamic vision problems at UBM [19, 20].

The experience gained in the fields of "satellite docking," "road vehicle guidance" and "on-board autonomous aircraft landing approaches" by machine vision, in the mid-1980s led to the conclusion that the joint use of dynamical models and temporal predictions for several aspects of the overall problem was key to achieving a quantum jump in the performance level of autonomous systems.

In addition to the state estimation for the physical objects observed and control computation based on these estimated states, it was the feedback of knowledge thus gained to image feature extraction and to the feature aggregation level, which allowed for an increase in efficiency of image-sequence evaluation of one to two orders of magnitude. (See Fig. 28.4 for a graphical overview.)

Following state prediction, the shape and the measurement models were exploited for determining:

- viewing direction control by pointing the two-axis platform carrying the cameras;
- locations in the image where information for the easiest, nonambiguous, and accurate state estimation could be found (feature selection);

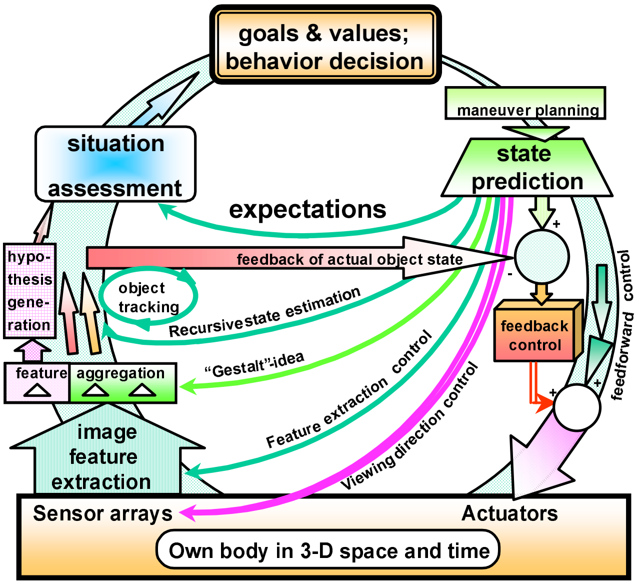

goals & values; behavior decision
maneuver planning
situation assessment
state prediction
expectations
hypo- thesis gene- ration
feedback of actual object state
object tracking
Recursive state estimation
feedback control
feedforward control
"Gestalt"-idea
feature aggregation
image feature extraction
Feature extraction control
Viewing direction control
Sensor arrays
Actuators
Own body in 3-D space and time

Figure 28.4: *Multiple feedback loops on different space scales for efficient scene interpretation and behavior control: control of image acquisition and -processing (lower left corner), 3-D "imagination"-space in upper half; motion control (lower right corner).*

- the orientation of edge features that allowed the reduction of the number of search masks and directions for robust yet efficient and precise edge localization;
- the length of the search path as function of the actual measurement uncertainty;
- strategies for efficient feature aggregation guided by the idea of the "Gestalt" of objects; and
- the Jacobian matrices of first-order derivatives of feature positions relative to state components in the dynamical models that contain rich information for interpretation of the motion process in a least squares error sense, given the motion constraints, the features measured, and the statistical properties known.

This integral use of

1. dynamical models for motion of and around the center of gravity taking actual control outputs and time delays into account;
2. 3-D shape models for specifying visually measurable features;
3. the perspective mapping models; and
4. prediction error feedback for estimation of the object state in 3-D space and time simultaneously and in closed-loop form

has been termed the "4-D approach." It is far more than a recursive estimation algorithm based on some arbitrary model assumption in some arbitrary subspace or in the image plane. It is estimated from a scan of recent publications in the field that even today most of the papers referring to "Kalman filters" do not take advantage of this integrated use of spatiotemporal models based on physical processes.

Initially, in our applications only the ego-vehicle has been assumed to be moving on a smooth surface or trajectory with the cameras fixed to the vehicle body. In the meantime, solutions to rather general scenarios are available with several cameras spatially arranged on a platform, which may be pointed by voluntary control relative to the vehicle body. These camera arrangements allow a wide simultaneous field of view, a central area for trinocular (skew) stereo interpretation, and a small area with high image resolution for "television". The vehicle may move in full 6 degrees of freedom; while moving, several other objects may move independently in front of a stationary background. One of these objects may be "fixated" (tracked) by the pointing device using inertial and visual feedback signals for keeping the object (almost) centered in the high-resolution image. A newly appearing object in the wide field of view may trigger a fast viewing direction change such that this object can be analyzed in more detail by one of the telecameras. This corresponds to "*saccadic*" vision as known from vertebrates and allows considerably reduced data rates for a complex sense of vision. It essentially trades the need for time-sliced control of attention and sampled-data-based scene reconstruction against a reduction in data rate of 1 to 2 orders of magnitude, as compared to full resolution in the entire simultaneous field of view.

The 4-D approach lends itself for this type of vision because both object-orientation and the temporal ("dynamical") models are available in the system already. This complex system design for dynamic vision has been termed **EMS-vision** (from **E**xpectation-based, **M**ultifocal and **S**accadic); it is actually being implemented with an experimental set of four miniature TV cameras on a two-axis pointing platform called "**M**ultifocal **a**ctive/**r**eactive **V**ehicle **Eye**," **MarVEye** [14].

In the rest of the chapter, major developmental steps in the 4-D approach over the last decade and the results achieved will be reviewed. As an introduction, in the next section a summary of the basic assumptions underlying the 4-D approach is given.

28.4.1 Basic assumptions underlying the four-dimensional approach

It is the explicit goal of this approach to take, as much as possible, advantage of physical and mathematical models of processes occurring in the real world. Models developed in the natural sciences and in engineering over the last centuries, in simulation technology and in systems engineering (decision and control) over the last decades form the basis for computer-internal representations of real-world processes:

1. the (mesoscopic) world observed occurs in *3-D space and time* as the independent variables; nonrelativistic (Newtonian) models are sufficient for describing these processes;
2. all interactions with the real world occur *"here and now,"* at the location of the body carrying special input/output devices; especially, the locations of the sensors (for signal or data input) and of the actuators (for control output) as well as those body regions with strongest interaction with the world (as, for example, the wheels of ground vehicles) are of highest importance;
3. *efficient interpretation of sensor signals* requires background knowledge about the *processes* observed and controlled, that is both its spatial and temporal characteristics. *Invariants* for process understanding may be abstract model components not obtainable at one point in time; similarly,
4. *efficient computation of* (favorable or optimal) *control outputs* can only be done taking complete (or partial) process models into account; control theory provides the methods for fast and stable reactions;
5. *wise behavioral decisions* require knowledge about the longer-term outcome of special feedforward or feedback control modes in certain situations and environments; these results are obtained from integration of the dynamical models. This may have been done beforehand and stored appropriately, or it may be done on the spot if analytical solutions are available or numerical ones can be derived in a small fraction of real time as has become possible now with the increasing processing power available. Behaviors are generated by triggering the modes available from step 4 in the foregoing.
6. *situations* are made up of arrangements of objects, other active subjects, and of the original goals pursued; therefore
7. it is essential to recognize *single objects and subjects*, their relative state, and for the latter also, if possible, their intentions in order to be able to make meaningful predictions about the future development of a situation (which is needed for successful behavioral decisions);
8. as the term *re-cognition* indicates, in the usual case it is assumed that the objects seen are (at least) generically known already. Only their appearance here (in the geometrical range of operation of the senses) and now is new; this allows a fast jump to an object hypothesis when first visual impressions arrive through sets of features. Exploiting background knowledge, the model-based perception proc-

ess has to be initiated. Free parameters in the generic object models may be determined efficiently by attention control and the use of special algorithms and behaviors;

9. in order to be able to do Step 8 efficiently, knowledge about "the world" has to be provided in the *context of "task domains"* in which likely co-occurrences are represented. In addition, knowledge about discriminating features is essential for correct hypothesis generation (indexing into the object database);
10. most efficient descriptions of objects (object classes) are usually done by *invariants in 3-D space* (for shape) *and time* (for motion constraints or stereotypical motion sequences). Modern microprocessors are sufficiently powerful to compute the visual appearance of an object under given aspect conditions in an image (in a single one, or even in several ones with different mapping parameters in parallel) at runtime. They are even powerful enough to numerically compute the Jacobian matrices for sensor/object pairs of features evaluated with respect to object state or parameter values; this allows a very flexible general framework for recursive state and parameter estimation. The inversion of perspective projection is thus reduced to a least squares model fit once the recursive process has been started. The underlying assumption here is that local linearizations of the overall process are sufficiently good representations of the nonlinear real process; for high evaluation rates such as video frequency (25 or 30 Hz) this is usually the case;
11. in a running interpretation process of a dynamic scene, *newly appearing objects* will occur *in restricted areas* of the image such that bottom-up search processes may be confined to these areas. Passing cars, for example, always enter the field of view from the side just above the ground; a small class of features allows detecting them reliably. *Subjects*, that is, objects with the capability of self-induced generation of control actuation, are characterized by typical (sometimes stereotypical, that is, predictive) motion behavior in certain situations. This may also be used for recognizing them (similar to shape in the spatial domain); note that many animal species recognize (dangerous) objects mainly through their motion behavior; and
12. the same object/subject may be represented internally at *different scales with various degrees of detail*; this allows flexible and efficient use in changing contexts (e.g., as a function of distance or degree of attention).

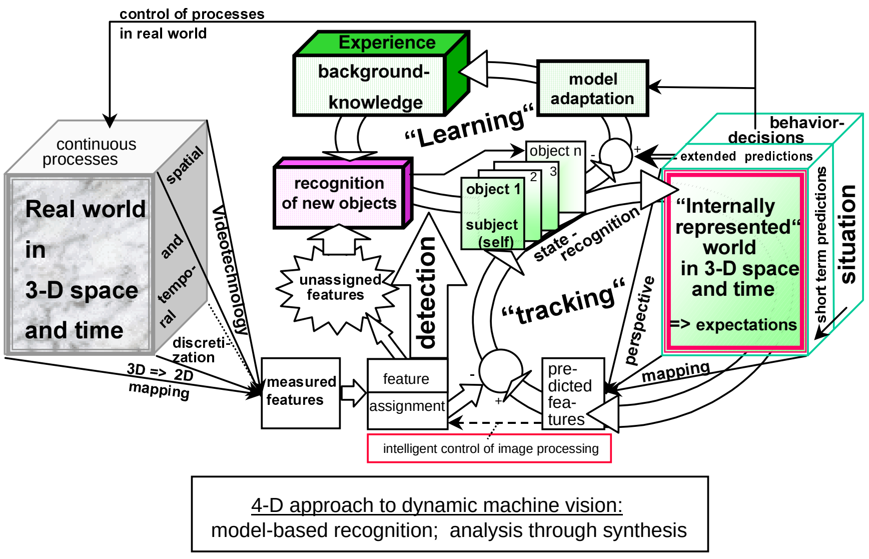


Figure 28.5: Survey on the 4-D approach to dynamic machine vision with three major areas of activity: object detection (central arrow upwards); tracking and state estimation (recursive loop in lower right); and learning (loop in center top). The latter two are driven by prediction error feedback.

28.4.2 Cybernetic structure of the four-dimensional approach

Figure 28.5 shows the three main activities running in parallel in an advanced version of the 4-D approach:

1. *Detection of objects* from typical collections of features not yet assigned to some object already tracked (center left, upward arrow). When these feature collections are stable over several frames, an object hypothesis has to be formed, and the new object is added to the list of those that have been regularly tracked (arrow to the right).
2. *Tracking of objects and state estimation* is shown in the loop to the lower right in Fig. 28.5; first, with the control output chosen, a single step prediction is done in 3-D space and time, the "imagined real world." This step consists of two components, a) the "where"-signal path concentrating on progress of motion in both translational and rotational degrees of freedom, and b) the "what"- signal path dealing with object shape. (In order not to overburden the figure these components are not shown.)
3. *Learning from observation* is done with the same data as for tracking; however, this is not a single step loop but rather a low frequency estimation component concentrating on "constant" parameters, or

it is even an off-line component with batch processing of stored data. This is currently a major activity in code development, which will help the architecture become more autonomous in new task domains as system experience grows. Both dynamical models (for the "where"-part) and shape models (for the "what"-part shall be learnable.

Another component under development but not detailed in Fig. 28.5 is situation assessment and behavior decision; this will be discussed in Section 28.4.8.

28.4.3 Generic four-dimensional object classes

The efficiency of the 4-D approach to dynamic vision is achieved by associating background knowledge about classes of objects and their behavioral capabilities in 3-D space and time with the data input. This knowledge is available in generic form; this means that structural information typical for object classes is fixed while specific parameters in the models have to be adapted to the special case at hand. Motion descriptions for the center of gravity (the translational object trajectory in space) and for rotational movements, both of which together form the so-called "where"-problem are separated from shape descriptions, called the "what"-problem. Typically, summing and averaging of feature positions is needed to solve the "where"-problem while differencing feature positions contribute to solving the "what"-problem.

Motion description. Possibilities for object trajectories are so abundant that they cannot be represented with reasonable effort. However, good models are usually available to describe their evolution over time as a function of the actual state, the control and the perturbation inputs. These so-called "dynamical models," usually, are sets of nonlinear differential equations ($\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}', t)$) with $\boldsymbol{x}$ as the n-component state vector, $\boldsymbol{u}$ as r-component control vector, and $\boldsymbol{v}'$ as perturbation input.

Through linearization around a nominal trajectory $\boldsymbol{x}_N(t)$, locally linearized descriptions are obtained which can be integrated analytically to yield the (approximate) local transition matrix description for small cycle times T

$$\boldsymbol{x}[(k+1)T] = \boldsymbol{A}\boldsymbol{x}[kT] + \boldsymbol{B}\boldsymbol{u}[kT] + \boldsymbol{v}[kT] \tag{28.1}$$

The elements of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are obtained from

$$\boldsymbol{F}(t) = \partial \boldsymbol{f}/\partial \boldsymbol{x}|_N \quad \text{and} \quad \boldsymbol{G}(t) = \partial \boldsymbol{f}/\partial \boldsymbol{u}|_N$$

by standard methods from systems theory.

Usually, the states cannot be measured directly but through the output-variables $\boldsymbol{y}$ given by

$$\boldsymbol{y}[kT] = h(\boldsymbol{x}[kT], \boldsymbol{p}, kT) + \boldsymbol{w}[kT] \tag{28.2}$$

where h may be a nonlinear mapping (see in the following), $\boldsymbol{p}$ are mapping parameters and $\boldsymbol{w}$ represents measurement noise. The measurements $\boldsymbol{y}$ contain the feature measurements in the 2-D image (like edge localization to subpixel accuracy) and, for the ego-body, data from inertial sensors.

On the basis of Eq. (28.1) a distinction between “objects” proper and “subjects” can be made: If there is no dependence on controls u in the model, or if this $\boldsymbol{u}(t)$ is input by another agent the term “object” is used, controlled by a subject in the latter case. If $\boldsymbol{u}[kT]$ may be activated by some internal activity within the object, be it by preprogrammed outputs or by results obtained from processing of measurement data, the term “subject” is chosen.

Shape and feature description. With respect to shape, objects and subjects are treated in the same fashion. Only rigid objects and objects consisting of several rigid parts linked by joints have been treated; for elastic and plastic modeling see [12, 21]. Because objects may be seen at different ranges, appearance may vary considerably in size. At large ranges the 3-D shape of the object is usually of no importance to the observer, and the cross section seen contains most of the information for tracking. However, this cross section depends on the angular aspect conditions; therefore, both coarse-to-fine and aspect-dependent modeling of shape is necessary for efficient dynamic vision. This will be discussed briefly for the task of perceiving road vehicles as they appear in normal road traffic.

- *Coarse-to-fine shape models in 2-D*: Seen from behind or from the front at a large distance, any road vehicle may be adequately described by its encasing rectangle. This is convenient as this shape has just two parameters, width b and height h. Absolute values of these parameters are of no importance at larger distances; the proper scale may be inferred from other known objects seen, such as road or lane width at that distance. Trucks (or buses) and cars can easily be distinguished. Experience shows that the upper boundary, and thus the height of the object, may be omitted without loss of functionality. Reflections in this spatially curved region of the car body together with varying environmental conditions may make reliable tracking of the upper body boundary very difficult). Thus, a simple U-shape of unit height (about 1 m turned out to be practical) seems to be sufficient until 1- to 2-dozen pixels can be found on a line crossing the object in the image. Depending on the focal length

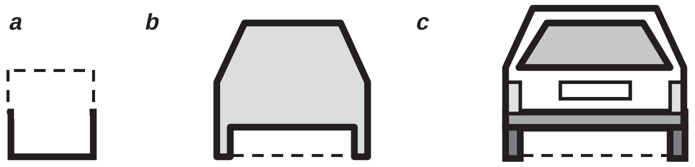


Figure 28.6: *Coarse-to-fine shape model of a car in rear view:* ***a*** *encasing rectangle (U-shape);* ***b*** *polygonal silhouette;* ***c*** *silhouette with internal structure.*

used, this corresponds to different absolute distances. Figure 28.6a shows this shape model. If the object in the image is large enough so that details may be distinguished reliably by feature extraction, a polygonal shape approximation as shown in Fig. 28.6b or even with internal details (Fig. 28.6c) may be chosen. In the latter case, area-based features like the license plate, the tires or the signal light groups (usually in yellow or reddish color) may allow more robust recognition and tracking.

If the view is from an oblique direction, the depth dimension (length of the vehicle) comes into play. Even with viewing conditions slightly off the axis of symmetry of the observed vehicle, the width of the car in the image will start increasing rapidly because of the larger length of the body and the sine-effect in mapping. It is usually impossible to determine the lateral aspect angle and body width and length simultaneously from visual measurements; therefore, switching to the body diagonal as a shape representation has proven to be much more robust and reliable in real-world scenes [22].

Just for tracking and relative state estimation, taking one of the vertical edges of the lower body together with the lower boundary of the object body has proven to be sufficient in most cases [23]. This, of course, is domain specific knowledge, which has to be introduced when specifying the features for measurement in the shape model.

In general, modeling of measurable features for object recognition has to be dependent on aspect conditions. Experience demonstrates that area-based features should play an important role in robust object tracking. Initially, this has been realized by observing the average gray value on the vehicle-side of edge features detected; with more computing power available, color profiles in certain cross sections yield improved performance. A special algorithm has been developed for this purpose. Figure 28.7 shows its application for tracking a car by locking onto intensity or color profiles; the results of the vertical and the horizontal cross sections are interpreted in conjunction, and the search stripes Vi, Hi position each other consecutively depending on the last results. If range changes, the ex-

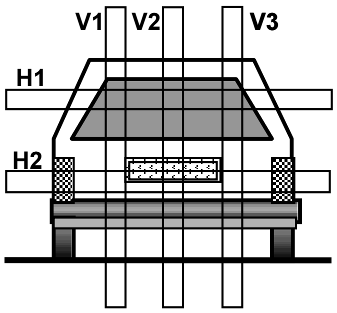


Figure 28.7: *Tracking of a car with intensity profiles in five cross sections (two horizontal and three vertical) centering each other mutually.*

tension of the profile also changes; the normalized profile, however, remains the same. This collection of profiles measured in a specific way is treated as a complex feature of this object under these aspect conditions. Provided that computing power is adequate, areas segmented with respect to color or texture will be the long-term solution.

- *Full 3-D models with different degrees of detail*: Similar to the 2-D rear silhouette, different models may also be used for 3-D shape. The one corresponding to Fig. 28.6a is the encasing box with perpendicular surfaces; if these surfaces can be easily distinguished in the image, and the separating edge can be measured precisely, good estimates of the overall body dimensions may be obtained from small image sizes. As space limits details here, the interested reader is referred to Schick [24], Schick and Dickmanns [25], and Schmid [22] where polyhedral models of road vehicles are treated.

28.4.4 Image feature extraction

Two classes of feature extraction algorithms are used: oriented edge features extracted by ternary mask correlation in horizontal or vertical search paths (a rather old component); and area-based segmentations of "stripes" of certain widths, arbitrarily oriented in the image plane (a new one, labeled T_{ij} in Fig. 28.8).

Intelligent control of the locations of application and of the most favorable parameters of these algorithms is essential for efficient tracking. In the 4-D approach, these parameters are set by predictions for the object state at the next sampling point in time based on the *spatiotemporal models*: the 3-D shape model, the relative position and orientation to the camera, and application of perspective mapping. From Fig. 28.8 it may be seen that a small percentage of image data properly analyzed allows objects to be tracked reliably and precisely when used

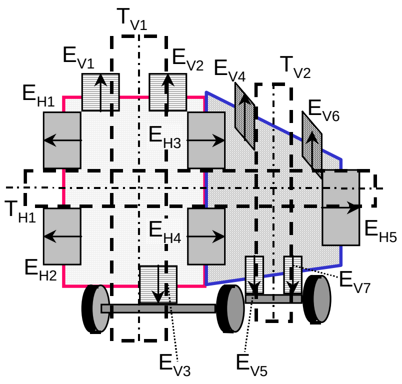


Figure 28.8: *Intelligent control of image feature extraction parameters in the algorithms CRONOS (for edges, marked with a window label E_{ij}) and "Triangle" (labeled T_{ij} , large rectangles with broken lines) for efficient object tracking and state estimation in the 4-D approach.*

in a tight bottom-up and top-down loop traversed frequently (25 Hz); this has to be seen in connection with the feedback loops of Fig. 28.4.

The information for sizing of the image areas to be evaluated and the search path lengths is derived from the statistics observed in the recursive estimation process.

K(C)RONOS for efficient edge feature extraction. Based on previous experience of [26, 27] for the "BVV"-image sequence processing systems (see Dickmanns and Graefe [28] for reference), for the transputer systems of the early 1990s a very general edge feature extraction routine KRONOS has been developed in the transputer language "Occam" [29]. With transition to general PC type processors this code has been translated to C and C++ and improved further; this version has been dubbed CRONOS [30].

Basic parameters of this efficient *edge detection* algorithm are the mask size m describing the width of the operator (3, 5, 9 and 17 pixels), and the angular orientation α (orientation index i) giving the direction of greatest sensitivity (see Fig. 28.9). For smaller mask widths, less angular orientations are possible. For $m = 17$, the direction index 0 corresponds to vertical, 16 to horizontal.

Search paths for edge detection are confined to horizontal and vertical in order to keep one (line or column) index constant; diagonal ones were also used initially, but they turned out to be superfluous. In summary, first a sum of m pixels along the mask orientation is computed, yielding an average gray value for the mask elements; second a ternary correlation is performed across n of these average gray values in a direction approximately perpendicular to the mask direction.

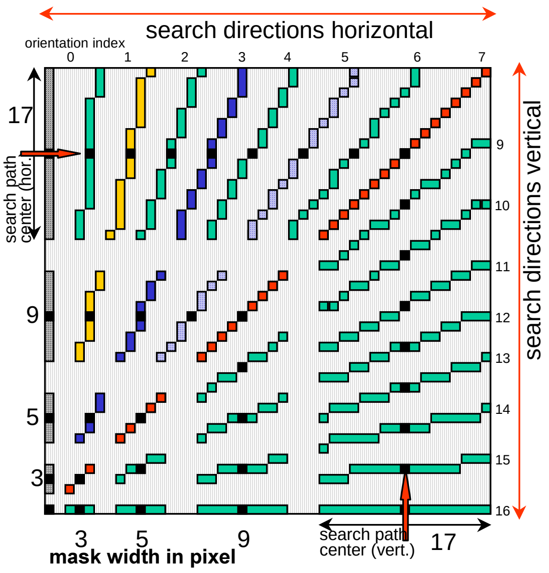

search directions horizontal
orientation index
0
1
2
3
4
5
6
7
search directions vertical
9
10
11
12
13
14
15
16
17
9
5
3
search path center (hor.)
search path center (vert.)
3
5
9
17
mask width in pixel

Figure 28.9: *Basic edge direction elements as used in K(C)RONOS; each element is "collapsed" to a single average value (central black dot).*

- *Column summation as lowpass filter operation in tangent direction*: First the sum of all pixel values along the direction of orientation is computed over the entire search window. This yields a single vector for any size of mask width. The corresponding averaged gray value is stored in the "base vector" of the search path, geometrically representing the central pixel array of the search path (marked as a black dot in Fig. 28.9). This is the reason for selecting mask widths m of size $2^k + 1$ with $k = 1$ to 4. From a spatial frequency point of view this corresponds to lowpass filtering along the mask direction over the width m pixel (with a distortion in length depending on the direction of orientation). However, normal to this direction full resolution is preserved; this fact is exploited for precise edge localization by specifying the edge direction to be used dynamically according to the predicted edge direction to be found for the (4-D) object tracked. This is part of the realization of the idea of "Gestalt" as discussed in psychology.
- *Mask parameter selection*: Depending on the parameters of the predicted edge in the image, the mask parameters will be chosen. Edges with imprecisely known directions and more strongly curved ones are detected with masks of small widths m (lower left in Fig. 28.9), almost straight ones with m large. By using several orientation directions on the same search path, the correlation results (see in what

follows) may be interpolated; this will allow finding the edge orientation with a precision of only a few degrees.

- *Ternary masks eliminate the need for multiplication*: Correlation masks are formed along the search path direction. By choosing the same correlation weight factor along the direction of mask orientation over width m, all sizes of masks can be handled just by starting from the summed base vector. By further specialization of the correlation weights equal to plus 1 in one half of the masks and minus 1 in the other half (one half of the "receptive fields", see Fig. 28.10), no multiplication is required in evaluating the correlation value. This value of a ternary mask is just equal to the difference of the sums of the pixel values in each field.

  This allows resorting to the base vector only, thereby making processing very efficient. "*Receptive fields*" for feature extraction, or in short just "*masks*" or "edge element templates," are defined by combining an equal number n_1 of the oriented columns of the base vector with positive and negative sign. A variable number n_0 of zero-weighted columns may be chosen in between, if desired. This yields the additional parameters for a "ternary correlation mask" or "edge template": number of columns n_1 set to $+1$ and to -1, and the number of central columns of pixels weighted with zero, that is dropped altogether. Total mask depth then is $n = 2 \times n_1 + n_0$; the mask of size $m \times n$ may be considered to be an orientation selective receptive field with total weight zero. Figure 28.10 shows a collection of receptive fields, their vector representations (marked with $+$, 0, $-$), and the search windows.

  If the position of an edge has to be determined, which is jagged on the pixel scale, such as, for example, the border between a track and spreading grass on a natural road, a larger number for n_0 is preferable. On a highway with well-kept lane markings, $n_0 = 1$ is a good choice; because of various reasons, in many real life scenes this value is a good default value. The limiting case $n_0 = 0$ has not been shown to be very practical. The value for n_1 should be chosen such that meaningful mask responses are obtained. If the boundary between two areas of texture with different average gray values has to be found, each half mask should be large enough as compared to the spatial frequency of the texture so as to average the gray value correctly. If a thin line has to be detected, n_1 should be small enough not to cover more than the width of the line. This clearly says that in images where the viewing direction is almost parallel to the ground, for example, for vehicles moving close to the ground, the scaling of mask size should go with one over distance of scene elements mapped, in general.

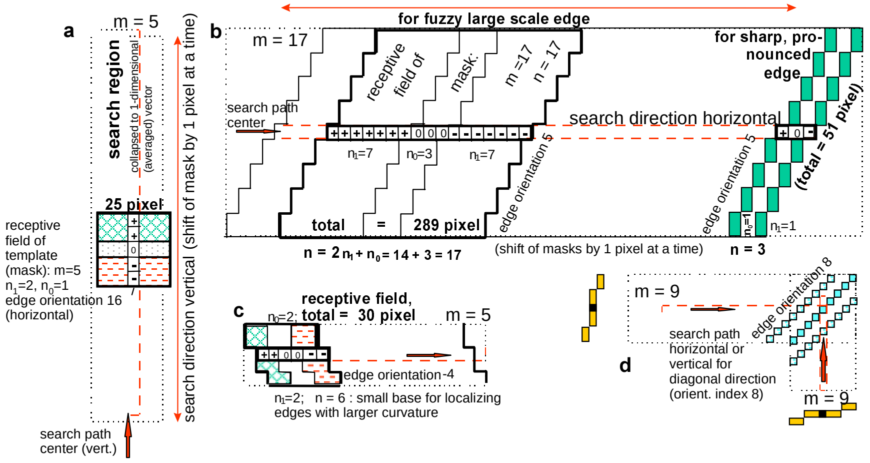

a
m = 5
search region
collapsed to 1-dimensional (averaged) vector
25 pixel
receptive field of template (mask): m=5 n1=2, n0=1 edge orientation 16 (horizontal)
search path center (vert.)
search direction vertical (shift of mask by 1 pixel at a time)
b
for fuzzy large scale edge
m = 17
receptive field of mask: m = 17 n = 17
search path center
n1=7 n0=3 n1=7
total = 289 pixel
n = 2n1 + n0 = 14 + 3 = 17
edge orientation 5
search direction horizontal
for sharp, pronounced edge
(total = 51 pixel)
n0=1 n1=1
n = 3
(shift of masks by 1 pixel at a time)
c
receptive field, n0=2; total = 30 pixel
m = 5
edge orientation -4
n1=2; n = 6 : small base for localizing edges with larger curvature
d
m = 9
search path horizontal or vertical for diagonal direction (orient. index 8)
edge orientation 8
m = 9

Figure 28.10: Representation of receptive fields by 1-D arrays (vectors).

In the *tracking* mode, the masks are evaluated along horizontal or vertical search paths (one pixel index kept constant) depending on the orientation of the edge such that the angle between mask- and search direction is as close as possible to 90°. This means that horizontal search paths are preferred for edge orientations between plus and minus 45° from the vertical, and vertical ones for edge orientations between plus and minus 45° from the horizontal (see lower right corner of Fig. 28.10).

For the reasons mentioned, mask orientation should be as close as possible to tangential to the edge direction that is to be determined. Only in this case is full resolution preserved. For the detection of previously unknown edges, several mask orientations should be tried. No other kind of image preprocessing needs to be done.

- *Monitoring image intensities*: For proper scaling and for specification of thresholds for image operations to follow, during the first algorithmic step of “transversal” averaging over the mask width, the maximal and minimal gray values along the search path length are monitored and stored.

  The difference of both values is an indication of the intensity dynamics in the (lowpass filtered) search region of the image. Performing the same monitoring procedure during gray-value summation for each pixel along the search path (i. e., each value of the base vector), the dynamics in intensity for all pixels may be evaluated. If the difference between maximal and minimal value in one mask column is too large, this is an indication that this operation may not be meaningful at all; other mask orientations should be tried in this image region. These checks are relatively costly; for this reason they are often left out in routine cases.

- *Some guidelines*: The parameters have to be adjusted depending on the type of edge to be found:
  - Low curvature edges call for large values of m and proper selection of mask orientation (e. g., based on previous results and proper prediction).
  - High curvature edges limit m to smaller values (or require higher resolution for the imaging process, e. g., larger focal length for the camera). Mask width m and the number of central zero columns n_0 should be selected such that the curved edge may completely fall into the 0 columns blended out; this will yield maximal mask response.
  - Steep intensity gradients and narrow edges call for small values of n_0; edges with lower transversal spatial frequency contents require larger values of n_0.
  - Large areas with a rather wide transition zone in between (e. g., dirt roads with a fuzzy boundary to the shoulders) are best detected with large values of n_0 and n_1; values of $n_0 = 3$ and $n_1 = 7$ with $m = 9$ or 17 depending on look-ahead distance have shown good results.
  - Narrow, sharp lines (a few pixel wide) require small values of n_0 and n_1 and proper orientation; if not known sufficiently well, several neighboring orientations should be tried in parallel. For poorly known edge orientation, it may be better to use small values of m, in addition, on three neighboring search paths. The edge locations obtained by extremal mask results may be combined for achieving the tangent line in the central search path (to subpixel accuracy). The best tangent direction is the line connecting the extremal values in the two neighboring paths (second-order fit), shifted, however, to pass through the location of the central extremal value.
- *Correlation results*: The most simple gradient operator with $n = 2$, $n_0 = 0$ and $n_1 = 1$ computes the difference of two neighboring base vector components along the search path. Figure 28.11 shows an example with $n_1 = 2$, $n_0 = 1$ and $m = 17$. It displays a typical situation; the extrema in mask response determine the position of the edge. Taking several neighboring values into account, this location may be determined to subpixel accuracy; for clean, almost straight edges, accuracies in the one-tenth pixel range are achievable. Several interpolation schemes have been investigated; a parabola, based on the extremal value and one neighboring value to each side only, seems to be an efficient solution in most cases.

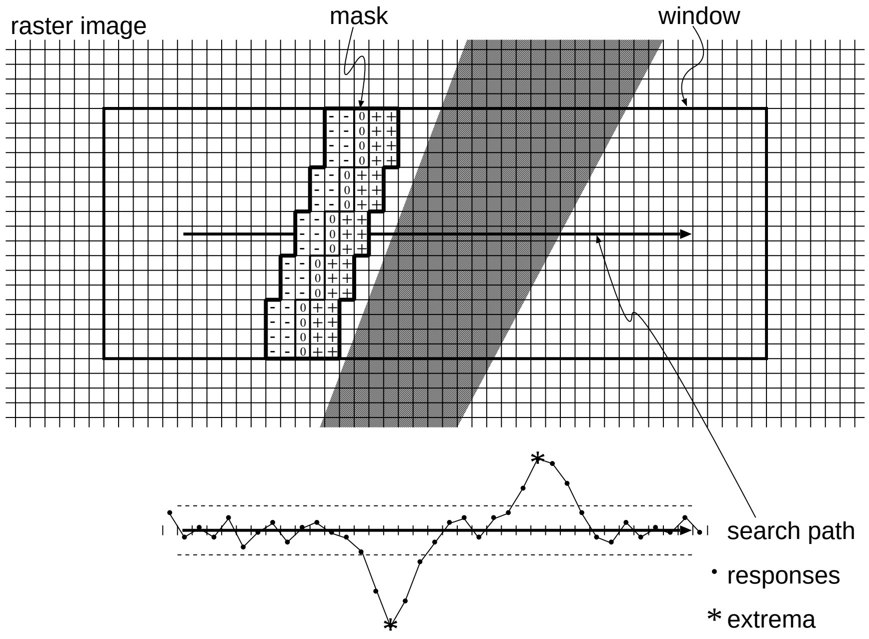


Figure 28.11: *Edge correlation with ternary masks; local extrema yield edge position.*

For larger values of n_1 all the plus and the minus columns of the first mask in the search window are added first, yielding some average gray value on each side. At the extremal value found, these values may be used for improving the association of this edge with an object known from previous evaluations, or with other edges from neighboring search paths. Shifting the mask in search direction by one pixel is then done by adding the new and subtracting the last vector component (or vice versa) in each field.

Working with masks of different depth n on the same search path, thus, may be done by very efficiently exploiting the sum terms for each column, that is, each vector component along the search path in multiple ways. In the next section it will be shown how this same vector may be used for fast area-based processing on multiple resolution scales along the search path.

By controlling search path position, direction, and length intelligently from the higher, object-oriented interpretation levels based on actual data evaluated in combination with knowledge about the process observed, very much improved efficiency in image sequence processing may be obtained. This is not on a percentage but rather on an order of magnitude level!

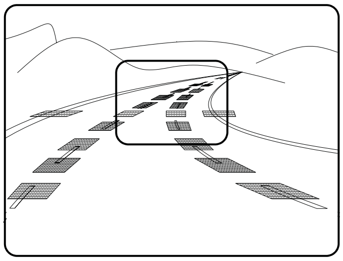


Figure 28.12: *Adaptation of parameters for edge detection and tracking according to knowledge from previous images and measured original perturbations, for example, pitching velocity sensed inertially.*

- *Comparison to wavelets*: *Wavelets* [31] are special local 2-D filters with nice theoretical properties. They come in different scales and with different directional orientations. The computational load to convolve these filters with the image is rather high. As compared to wavelets with smooth transitions between rational weighting factors depending on their offset from the center of the field, the “box-like” ternary weighting schemes are rather crude, but compact in code and very efficient. No multiplication operations are necessary. With the parameterizations discussed in the preceding text, the software package CRONOS allows qualitatively comparable results at orders of magnitude less computing load.

  Feeding these coarse results into a high-frequency recursive estimation process with good smoothing properties has shown to satisfy all practical requirements. It is assumed that the short cycle times more than offset the disadvantages of the box-like filter type. To date, time has not allowed careful comparison of the methods and an assessment of the relative performance advantages.
- *Application examples of CRONOS*: In Fig. 28.8 the areas labeled E_{ij} show typical distributions of search areas and mask orientations for tracking edges on a rigid object with different shades on the sides. Because the average gray values of each region on the object are known from previous measurements, the search paths should be commanded from the center of the region towards the outside. Usually, the background behind a moving object is much more variable than the surface brightness of the object; of course, this is not true when moving through shades of trees. When searching in the inside-out direction, the average gray value of the trailing mask section may be used for solving the correspondence problem for boundary features of this same face.

  The *search windows* are positioned in the image plane after state prediction and perspective mapping such that sufficiently large masks with proper parameters for robust feature detection (e. g., width, orientation, and number of columns) can be applied. Areas crowded with features, such as is the case both above the axles and the wheels are deliberately avoided as confusion of features may cause more harm than good in the overall interpretation process. Because a redundant set of features is usually used, the additional information to be gained by uncertain features may not be worth the effort. This conservative attitude pays off mostly in having the unspent computing power available to invest at other locations in the image for extracting less ambiguous features. Figure 28.12 shows an example of *road lane recognition*. Due to road curvature recognition (see previous pictures), and to the ego-motion predicted, the search windows are positioned along a curve in the image plane. The size of the search windows is specified depending on the prediction errors experienced previously. Because of the range effects associated with lines positioned higher in the image when looking almost parallel to the road surface, mask size is adjusted according to range. Note that mask orientation is different for each individual lane marking at the same range. This change in orientation is derived from the known width of the lane and elevation of the camera above the ground; it is computed for each mask at each step of the recursive iteration process. The positions of the extremal values in mask correlation (see Fig. 28.11) are taken as edge center point and form the elements of the measurements y in Eq. (28.2).

Area-based feature extraction: The "triangle"-algorithm. If there is texture on the surface (either from design or through shadows from trees in sunshine), edge-based methods may saturate rather easily. Dominant color components or certain texture patterns may nonetheless allow area-based tracking of the object. Computing power is not yet available to do this at video rate over the entire image to a sufficient extent; affordable solutions may still be a decade away. Therefore, an intermediate step has been used so as to take advantage of some area-based information to be obtained easily by combining the first step of column summation in edge feature extraction (already discussed here) with multiple scale representations in just one dimension.

The basic underlying assumption is that meaningful results can be achieved for area-based processing when the 2-D problem in the image plane is reduced to a small set of 1-D ones, the selection of which is controlled by deeper insight into the continuity conditions of the problem at hand. If there were sufficient computing power available at no cost, this approach would be superfluous; however, in a world of limited resources and with sensitivity to costs it may be worthwhile even in the long run because it allows for the most efficient use of the invested components.

- *Parameter selection*: Beside horizontal and vertical "test stripes" as already discussed here in connection with the edge extraction algorithms K(C)RONOS (called "search windows" or "-paths" there), any stripe along a line connecting two points in the image may be used as a "triangle base." Stripe width is denoted by w (corresponding to m in CRONOS), and is collapsed first into a single summed (or averaged) vector component between the limiting points P_i (see Fig. 28.13). Depending on the inclination of this line, summing (or averaging) is done in pixel rows or columns such that the angle between line inclination and rows or columns is as close as possible to 90°; Fig. 28.13 shows corresponding cases. Here, stripe width summation is done with one index fixed, while selection of the location of its center point is done according to stripe orientation with a Bresenham algorithm [32].

  Selection of the stripe width w has to be done according to the problem at hand: If large homogeneous areas are looked for, w should be large, corresponding to lowpass filtering normal to the stripe direction. For color processing or large noise amplitudes and small w, median operations may be better suited than averaging; this is also true for the computation of pixel values on the triangle levels (see below). Stripe P1-P2 (shown $w = 3$ pixel wide in Fig. 28.13, lower part) may be used for determining an intensity or color profile across an image section of special interest. For tracking a truck under an oblique view, segmentation of homogeneous regions may be advantageous (see Fig. 28.8, stripes designated with T_{ij} for the rear and the side face). By commanding the search stripe directed orthonormal to the center of detected homogeneous areas in a cross-wise manner, the center of the homogeneous area will be tracked.

  If a longitudinally extended intensity or color blob has to be tracked (see upper part of Fig. 28.13) and the angular orientation has to be determined roughly, the three stripes defined by P3 to P8 in Fig. 28.13 will yield good results. They are positioned such that each center is approximately at the center of the blob's cross section detected last time by the orthonormal stripe (or of the predicted one). In order to be able to track the angular orientation of the major skeletal line of the body, two stripes are positioned approximately normal to this direction at about 1/3 and 2/3 of the body extension. Connecting the centers of the homogeneous areas measured actually yields the new angular orientation in the image. This also specifies the direction for the stripe covering the longitudinal extension of the body in the image.

  Typical feature elements for the measurement vector y would then be the location of the blob center and its angular orientation (maybe also its size).

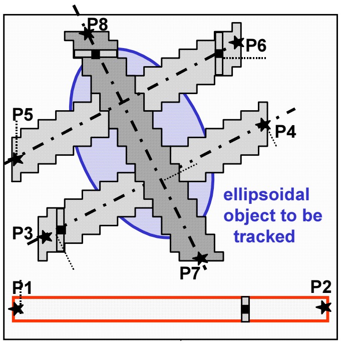


Figure 28.13: *Definition of the base-vector for the "triangle algorithm" by any two points: $*$: two points defining a triangle base (stripe); dash-dotted line: skeletal line of triangle (width: 1 pixel; its inclination defines a length component!); shaded bars: vertical or horizontal stripe width (any odd number of pixels 3, 5, 7, ...), collapsed into a single point ■ for efficient handling; ■: vector element representing (averaged) stripe width; all these elements between and including the defining points form the triangle base (preferably 2^n pixel) above which the lower resolution levels (reduced by a factor of 2) are computed.*

- *Construction of the triangle representation*: Figure 28.14 shows the multiple-scale representation levels in an arrangement visualizing the name of the method selected. It may be considered as a reduced (1-D) version of the well-known "pyramid"-technique [33]. Cross sections through larger blobs may be found efficiently on higher levels of the triangle by intelligently selected "cuts" without distraction by spurious details as often occurs on lower levels.

  During computation of the actual values on the various representation levels, maximum and minimum pixel intensities may be monitored and stored on each level; together with statistical evaluations of intensity differences between neighboring pixels this yields valuable information for scene interpretation and threshold adaptation. Intensity or color profiles for cross sections of objects characterize the object. Figure 28.7 has shown one arrangement for robust tracking of a car. In Fig. 28.8, the three triangle bases T_{V1}, T_{V2} and T_{H1} allow to track the dominant regions in the visual appearance of the truck trailer. Details of textures may be averaged away on a proper scale while on another scale a certain intensity or color profile may allow internal structure to be recognized.

  The extremal values of pixel intensities on some triangle levels may be exploited for controlling threshold values in the edge feature extraction algorithm K(C)RONOS. The advantages of cross-feeding information between these two algorithms have not yet been fully used; a knowledge base for certain task domains and situations is

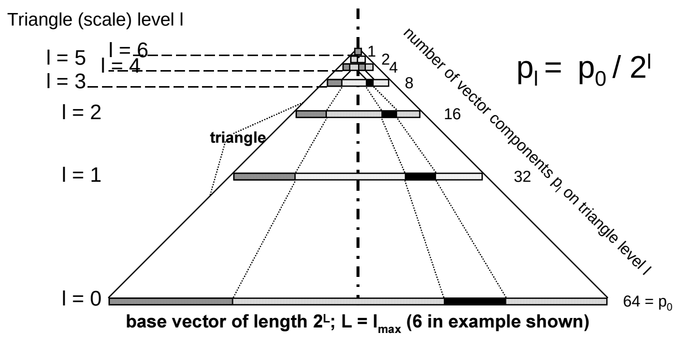

Triangle (scale) level l
l = 6
l = 5
l = 4
l = 3
l = 2
l = 1
l = 0
triangle
1
2
4
8
16
32
64 = p0
number of vector components pl on triangle level l
pl = p0 / 2^l
base vector of length 2^L; L = lmax (6 in example shown)

Figure 28.14: *Multiple-scale representation of a stripe by a triangle (fits into vector of length $2p_0$).*

being accumulated and will be integrated in the new system under development.

For *road recognition*, tracking of lanes usually ends when the widths of lane markings in the image drop below 1 to 2 pixels; with the bifocal camera arrangement used up to now this is in the 100-meter range. Neglecting lane marks altogether at greater distances and concentrating on the center of the road by using the triangle method described, robust road tracking becomes possible to much larger distances.

Because lateral offsets from the local tangent direction to the road go with the second and third power of look-ahead range, the errors in curvature due to unknown numbers of lanes and imprecise road width measurements are relatively small.

Because of the inverse length mapping with distance in perspective projection, efficient image interpretation is done on different triangle levels as a function of distance. Nearby, larger widths w of stripes in combination with working on higher triangle levels will increase efficiency. Farther away, smaller values of w and working on lower triangle levels will yield sufficient resolution.

In addition, under normal conditions color processing may allow good discrimination between grass and dirt road where the differences in intensity are rather small, thereby improving robustness in road recognition. This, of course, depends very much on the time of year and other factors; in any case, area-based processing allows to decrease the number of cases where a combined system may fail.

Figure 28.15 shows the detection of a *road fork* by several transversal stripes selected normal to the road driven (Fig. 28.15a). The intensity profiles (Fig. 28.15b) first show a widening of the homogeneous areas, then a separation into two plateaus (SB2 and SB3) with an increasing gap in between. The interpretation level takes this as

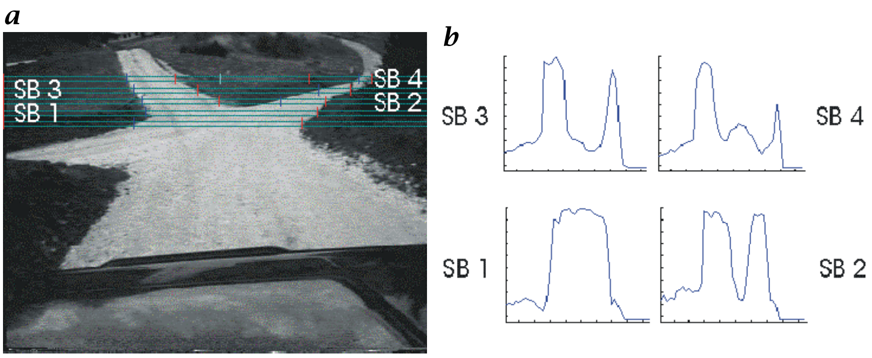


***Figure 28.15:** Detection of a road fork by the triangle algorithm.*

an indication that two separate roadways are emergent. It would command new search stripes next time, one set to the left (continuing to be horizontally oriented) and one to the right, which would be oriented approximately parallel to the old right road boundary but displaced to the right. Each one will be tracked now as a separate object with specific parameter adaptations.

28.4.5 Feature aggregation

If working with edge features only, there may be several features that create a meaningful combination representing some part of an object known to the system under some aspect conditions. This is the frequently cited combinatorial explosion problem of feature aggregation. Left alone with just edges when starting from scratch, the correspondence problem between features extracted in the image and parts of objects in the database may well be practically unsolvable in the general case.

However, when adjacent gray or color values can be taken into account the problem often is much less hard. For this reason, the average gray value of one field of the edge template at the edge position is returned to the interpretation process as one component of the result in K(C)RONOS. This allows grouping of all neighboring edge features of a homogeneous area in the image. In addition, properties of “good neighborhood” like colinearity, smooth curvature, appropriate length ratios of feature groupings, or parallel orientation may help in reducing the correspondence problem during hypothesis instantiation. The triangle algorithm may contribute information on more extended homogeneous areas in regions of interest.

During tracking, the situation is much alleviated once a good model for shape and motion has been found. Now, the appearance of features can be predicted, and parameters for the feature extraction algorithms (like mask size and orientation) may be selected in an optimal fashion. The variances in the estimation processes give an indication of prediction uncertainty; this is exploited to confine the search range and thus improve efficiency. If features come to be too close to each other in the image (say one to three pixel) and the danger of confusion is likely, these features should be discarded altogether because wrong feature correspondence is a major source of error in the tracking process.

For newly detected objects it usually does not make sense to test too many object hypotheses in parallel based on a single image (always consisting of shape and aspect conditions as two independent components). As perturbations will change from one image to the next, and because dynamically moving objects will change aspect conditions in a systematic way it will, in general, be advantageous to start early tracking of the most likely hypothesis. If computing power allows, maybe a few most likely hypotheses for the task domain under scrutiny should be started in parallel and be tracked with low cycle times. In addition to shape information, knowledge about process development over time is a powerful discrimination component. Short cycle times reduce the search range and alleviate the correspondence problem, a fact often overlooked in early AI-approaches.

28.4.6 State estimation

The basic approach has been described many times (see Wünsche [19, 20], Mysliwetz [27], Dickmanns and Graefe [28]) and has remained the same for visual relative *state estimation* for more than a decade. Only a basic outline will be given here.

However, in order to be able to deal more effectively with the general case of scene recognition under (more strongly) perturbed ego-motion, an inertially based component has been added [34, 35].

This type of state estimation is not new at all if compared to inertial navigation, for example, for missiles; however, here only very inexpensive accelerometers and angular rate sensors are being used. This is acceptable only because the resulting drift problems are handled by a visual state estimation loop running in parallel, thereby resembling the combined use of (relatively poor) inertial signals from the vestibular apparatus and of visual signals in vertebrate perception. Some of these inertial signals may also be used for stabilizing the viewing direction with respect to the stationary environment by direct negative feedback of angular rates to the pointing device carrying the cameras. This feedback actually runs at very high rates in our systems (500 Hz, see Schiehlen [13]).

- *Inertially based ego-state estimation* (IbSE): The advantage of this new component is 3-fold: 1. Because of the direct encoding of accelerations along, and rotational speed components around body fixed axes, time delays are almost nonexistent. These components can be integrated numerically to yield predictions of positions. 2. The quantities measured correspond to the forces and moments actually exerted on the vehicle including the effects of perturbations; therefore, they are more valuable than predictions from a theoretical model disregarding perturbations which are generally unknown. 3. If good eigenbehavior models are available, the inertial measurements allow estimating parameters in perturbation models, thereby leading to greater understanding of environmental effects.
- *Dynamic vision exploiting 4-D models*: With respect to *ego-state recognition*, vision now has reduced but still essential functionality. It has to stabilize long-term interpretation relative to the stationary environment, and it has to yield information on the environment, like position and orientation relative to the road and road curvature in vehicle guidance, not measurable inertially. With respect to other vehicles or obstacles, the vision task also is slightly alleviated because the high-frequency viewing direction component is now known; this reduces search range required for feature extraction and leads to higher overall system efficiency. These effects can only be achieved using *spatiotemporal models* and perspective mapping, because these items link inertial measurements to features in the image plane. With different measurement models for all the cameras used, a single object model and its recursive iteration loop may be fed with image data from all relevant cameras. *Jacobian matrices* now exist for each object/sensor pair; if two cameras with different focal length map the same object, two Jacobian matrices are needed.

The nonlinear measurement Eq. (28.2) is linearized around the predicted nominal state $\boldsymbol{x}_N$ and the nominal parameter set $\boldsymbol{p}_N$ yielding (without the noise term)

$$\begin{aligned} \boldsymbol{y}[kT] &= \boldsymbol{y}_N[kT] + \boldsymbol{\delta y}[kT] \\ &= \boldsymbol{h}(\boldsymbol{x}_N[kT], \boldsymbol{p}_N, kT) + \boldsymbol{C}_x\boldsymbol{\delta x} + \boldsymbol{C}_p\boldsymbol{\delta p} \end{aligned} \tag{28.3}$$

where $\boldsymbol{C}_x = \partial\boldsymbol{h}/\partial\boldsymbol{x}|_N$ and $\boldsymbol{C}_p = \partial\boldsymbol{h}/\partial\boldsymbol{p}|_N$ are the Jacobian matrices with respect to the state components $\boldsymbol{x}$ and the parameters $\boldsymbol{p}$ involved. Because the first terms to the right-hand side of the equality sign are equal by definition, Eq. (28.3) may be used for determining $\boldsymbol{\delta x}$ and $\boldsymbol{\delta p}$ in a least squares sense by a model fit. Assuming that differences $\boldsymbol{\delta y}[kT]$ between predictions by the dynamical model Eq. (28.1)) together with the forward projection model Eq. (28.2) and the actual measurements are small, linear relations should be sufficient for determining the unknowns $\boldsymbol{\delta x}$ and $\boldsymbol{\delta p}$ from $\boldsymbol{\delta y}$ as the prediction error measured (observability given; for the notion of observability see Maybeck [36] or Kailath [37]). This is the core of recursive estimation. A numerically stabilized version of the extended Kalman filter is used with $\boldsymbol{UD}/\boldsymbol{U}^T$-factorization and sequential innovation (see Wünsche [19, 20], Maybeck [36].

Note that with this dynamical formulation the inversion of the nonlinear perspective mapping equations has been transformed into a least-squares model-and-state adaptation. The core element for successful functioning of the approach is that the elements of the Jacobian matrices do not introduce singularities. These elements tell how a specific state variable or a parameter in the model affects the feature position in the specific image. While the exact check for *observability* is a bit more complex, a simple rule is as follows: A parameter or (set of) state component(s), which is linearly independent from other observable parameters or state components, is not (or poorly) observable, if all corresponding elements of the Jacobian are zero (or comparatively small). Then this parameter or state component cannot be iterated meaningfully with the features available. A typical example is the length parameter of a rectangular box looked at in length direction, where all features possibly yielding length information are self-occluded. In cases like this, the corresponding component should not be iterated but kept constant at the predicted value; hopefully, the aspect conditions will change such that later on this variable also may be iterated and improved again.

However, during the time this variable is unobservable, other state components or parameters can still be iterated successfully. This is another fundamental advantage of the approach chosen.

With computing power increasing rapidly, the cumbersome derivation of analytical expressions for the Jacobian elements has been abandoned, and the elements are now determined by proper numerical differencing of terms obtained from forward projection of perturbed states. This is especially useful for flexible handling of newly appearing or disappearing objects in connection with more general program code (see Dickmanns [29]).

28.4.7 Situation assessment

For each object an estimation loop is set up to yield best estimates for the relative state to the ego-vehicle including all spatial velocity components. For stationary landmarks, velocity is, of course, the negative of ego-speed. As this is known reliably from conventional measurements, the distance to a landmark of unknown size can be determined even with monocular vision exploiting motion stereo [23, 38, 39]. With all this information available from the surrounding environment and the most essential objects in it, an interpretation process can evaluate the situation in a task context and come up with a conclusion as to whether to proceed with the behavioral mode running or to switch to a different mode. Figure 28.22 (see Section 28.6.2) shows a typical scene tree with objects of relevance for a landing approach. Each branch of the tree corresponds to a coordinate transformation, the arguments of which have to be determined by measurements and state estimation, many of them by vision.

Fast in-advance simulations exploiting dynamical models and alternative stereotypical control activation yield possible alternatives for the near-term evolution of the situation. By comparing the options or by resorting to precomputed and stored results, these decisions are made. More details on this developing field are given in Dickmanns [40].

28.4.8 Generation of behavioral capabilities

Dynamic vision is geared to closed-loop behavior in a task context; the types of behavior of relevance, of course, depend on the special task domain. The general aspect is that behaviors are generated by control output. There are two basically different types of control generation:

1. Triggering the activation of (generically) stored time histories, so-called *feedforward control*, by events actually observed, and
2. Linking of actual control output to the difference between desired and actual state of relevant systems, so-called feedback control.

In both cases, actual control parameters may depend on the situation given. A very general method is to combine the two given in the preceding (as a third case in the list), which is especially easy in the 4-D approach where dynamical models are already available for motion understanding. The general feedforward control law in generic form is

$$\boldsymbol{u}(\tau) = \boldsymbol{g}(\boldsymbol{p}_M, \tau_M), \quad \text{with} \quad 0 \le \tau = t - t_{\text{Trig}} < \tau_M \tag{28.4}$$

where p_M may contain averaged state components (like speed). A typical feedforward control element is the steer control output for lane change of a car.

The general state *feedback control* law is

$$\boldsymbol{u}(\tau) = -\boldsymbol{K}^T \boldsymbol{\delta x}(\tau) \tag{28.5}$$

with $\boldsymbol{K}$ an $r \times n$ gain matrix. The gain coefficients may be set by pole placement or by a Riccati design (optimal linear quadratic controller) well known in control engineering [37]. Both methods include knowledge about behavioral characteristics along the time axis: While pole placement specifies the eigenvalues of the closed loop system, the Riccati design minimizes weighted integrals of state errors and control inputs.

The simultaneous use of dynamical models for both perception and control and for the evaluation process leading to behavior decision makes the 4-D approach so efficient. All that is needed for mission performance of any specific system then is a sufficiently rich set of feedforward and feedback behavioral capabilities. These have to be activated in the right sequence such that the goals are achieved in the end.

For this purpose, the effect of each behavioral capability has to be represented on the upper decision level by global descriptions of their effects:

1. For feedforward behaviors with corrective feedback superimposed it is sufficient just to represent initial and final conditions including time needed; note that this is a quasi-static description as used in AI-methods. This level does not have to worry about real-time dynamics, these are taken care of by the lower levels. It just has to know in which situations these behavioral capabilities may be activated with which parameter set.
2. For feedback behaviors it is sufficient to know when this mode may be used; these reflex-like fast reactions may run over unlimited periods of time if not interrupted by some special event. A typical example is lane following in road vehicle guidance; the integral of speed then is the distance traveled, irrespective of the curvatures of the road. These values are given in information systems for planning, like maps or tables, and can be used for checking mission progress on the upper level.

In addition, to some extent behaviors may be modified in order to disambiguate visual perception problems. For example, lateral position in the lane may be adjusted so as to allow vision past the vehicle in front.

28.5 Multiple loops in dynamic scene understanding

In order to be efficient locally on the one side and to keep track of situations in larger contexts on the other side, *multiple scales* are applied both in space and in time. One essential component in space is the multifocal camera arrangement and corresponding interpretations with models covering the entire spectrum. On different hierarchical levels in the system, emphasis is placed on different resolution levels; in general, the higher up the level, the broader is the region covered, and the less are details tracked.

A central issue in such systems is where to direct attention in which sequence. Unforeseen events have to be detected not only in the small region of actual special attention, but everywhere if they are relevant for mission performance. Because perceptual capabilities for a long time to come will not be sufficient for covering the entire environment simultaneously, background knowledge on the task domain at hand has to be used for intelligent control of the sparse resources available. Little is known as to how this is achieved in biological systems. Because of their complexity it is hard to obtain proven results. However, it is this capability that makes biologic systems continue to be superior to technical ones.

28.5.1 Flexibility and efficiency from intelligent control

In biological neural nets there is a large amount of feedback (not yet well understood) from higher interpretation levels to the image processing levels [41, 42]. Artificial neural nets have almost completely disregarded this fact up to now. One of the main goals of the 4-D approach was to provide a sound basis for this feedback from the higher levels onto image processing in order to cut down the computing power required. This is why, right from the beginning, no general-purpose abstract vision system has been looked for but rather "ecologically embedded" systems for certain task domains. Dynamic vision is embedded in closed-loop behavior; this seems to be essential for steady model adaptation and testing. Hopefully, by the time the base of experience is sufficiently large, there will also be sufficient computing power and room for abstraction available for "more general" systems. There are strong indications that this is happening right now.

Figure 28.2 has given an indication as to the abstraction levels used in space and time. Single objects form the pivotal point for storing knowledge about the world and, later on, for associating this knowledge with feature groupings appearing in the image data stream. Only by organizing this problem according to task domains and situations during task execution can the tendency for combinatorial explosion be kept manageable. Temporal constraints on the speed of evolution are also very essential for dynamic scene understanding. It is the steady development of actions and the relatively high sampling frequency provided by vision, which makes dynamic scene understanding possible, a fact overlooked in the initial AI-approaches. The temporal models of the 4-D approach in combination with time integrals over various time spans allow efficient handling of the complexity involved.

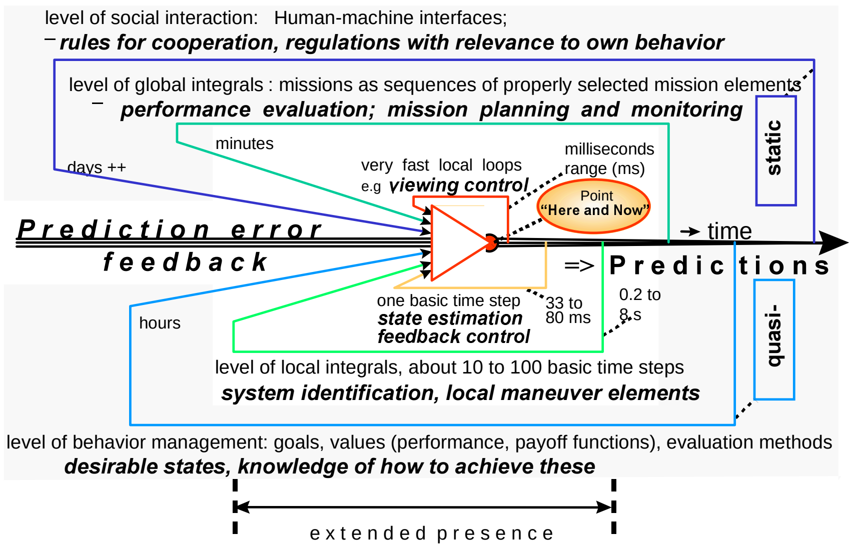

level of social interaction: Human-machine interfaces;
– rules for cooperation, regulations with relevance to own behavior
level of global integrals : missions as sequences of properly selected mission elements
– performance evaluation; mission planning and monitoring
static
minutes
days ++
very fast local loops
e.g viewing control
milliseconds range (ms)
Point "Here and Now"
Prediction error feedback
→ time
=> Predictions
one basic time step
state estimation
feedback control
33 to 80 ms
0.2 to 8 s
hours
quasi-
level of local integrals, about 10 to 100 basic time steps
system identification, local maneuver elements
level of behavior management: goals, values (performance, payoff functions), evaluation methods
desirable states, knowledge of how to achieve these
extended presence

Figure 28.16: Multiple feedback loops on different time scales in the 4-D approach and corresponding levels.

28.5.2 Multiple-loop closure on different time scales

The principles discussed here have led to parallel realizations of multiple loops in the interpretation process both in space and time; Fig. 28.4 has displayed the spatial aspects. In the upper half of the figure, the essential scales for feedback loops are the object level, the local situation level, and the global mission performance level on which behavior decisions for achieving mission goals are being done (see Fig. 28.2 also).

These decisions may be based on both local and extended predictions of the actual situation and on knowledge about behavioral capabilities of the ego-vehicle and of other subjects in the scene. The multiple loops used in our system in the time domain are displayed in Fig. 28.16. They range from the millisecond scale for inertial viewing direction control to several hours for ground and flight vehicles on the mission scale encompassing sequences of maneuvers, typically in the 1–10s range and feedback behavioral modes (with no direct restriction in time).

In the past, human operators and software developers have closed the two outermost loops labeled "quasi-static" in the figure. These are being tackled now for automation of the system structure being developed. It is felt that a unified approach encompassing techniques from systems dynamics, control engineering, computer simulation, and animation as well as methods from AI has become feasible.

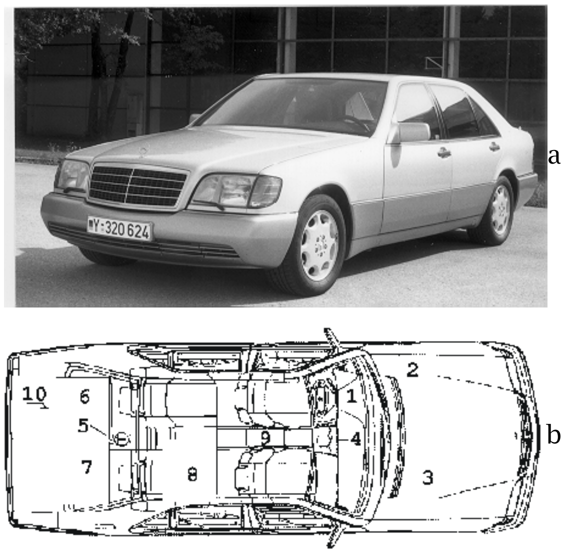


Figure 28.17: *The autonomous vehicle* ***VaMP*** *of UBM;* ***a*** *VaMoRs-P (VaMP);* ***b*** *top view of VaMoRs-P, components for autonomous driving: (1) electrical steering motor; (2) electrical brake control; (3) electronic throttle; (4) front pointing platform for CCD cameras; (5) rear pointing platform for CCD cameras; (6,8) Transputer image processing system; (7) platform and vehicle controllers; (8) electronics rack, human interface; (9) accelerometers (3 orthogonal); (10) rate gyros.*

28.6 Experimental results

28.6.1 Visual guidance of road vehicles

The autonomous road vehicles **VaMP** (see Fig. 28.17) and its twin **VITA II** of Daimler-Benz have shown remarkable performance in normal freeway traffic in France, Germany and Denmark since 1994. The **VaMP** has two pairs of bifocal camera sets of focal lengths 7.5 and 24 mm; one looks to the front (see Fig. 28.3), the other one to the rear. With 320×240 pixels per image this is sufficient for observing road and traffic up to about 100 m in front of and behind the vehicle. The vision system consisted of 46 transputers for image processing. The **VaMP** was able in 1994 to recognize road curvature, lane width, the number of lanes, the type of lane markings, its position and attitude relative to the lane and the relative state of up to 10 other vehicles including their velocity components. In each of the four video-data streams three vehicles have been tracked (see what follows and Fig. 28.19).

At the final demonstration of the EUREKA-project Prometheus in October 1994 near Paris, **VaMP** demonstrated its capabilities of free

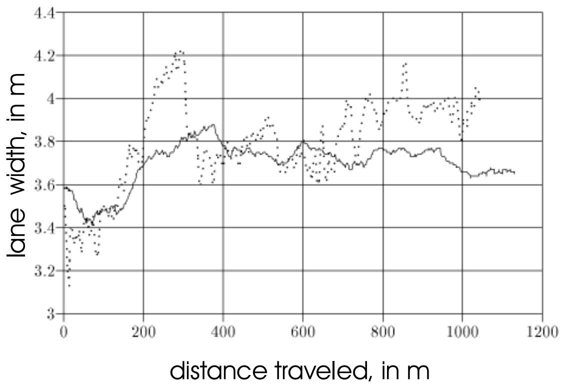


Figure 28.18: *Results for lane-width estimation with the same data with (solid line) and without (dotted line) modeling vehicle pitch motion as a damped second-order dynamic system.*

lane driving and convoy driving at speeds of up to 130 km/h in normally dense three-lane traffic. Lane changing for passing and even the decision as to whether *lane changes* were safely possible was done autonomously. The human safety pilot (driver) just had to check the validity of the decision and give a go-ahead input.

- *Road recognition* has been discussed in connection with the feature extraction algorithms; for more details see [27, 43]. The position and orientation of the vehicle relative to the lane is obtained simultaneously with the road and lane parameters because these variables affect each other through the perspective mapping process. For constant lane width, both horizontal and vertical curvature may be estimated. If lane width is variable and there exists vertical curvature (nonplanar environment), the interpretation of short image sequences may be ambiguous. In these cases, observability of the road parameters is given only when sufficiently extended data integration over distance is possible.

  Dynamical vehicle motion in pitch due to uneven ground (high spatial frequency components present) may also affect the quality of road parameter estimation even on a road that is flat in the average (low spatial frequency amplitude equal to zero). Figure 28.18 shows experimental results for lane width estimation with **VaMP** on a flat bumpy road with and without modeling the resulting vehicle motion in pitch. It is seen that taking the dynamic motion into account results in more smooth estimates for lane width. Even better results may be achieved if the camera viewing direction were inertially stabilized.

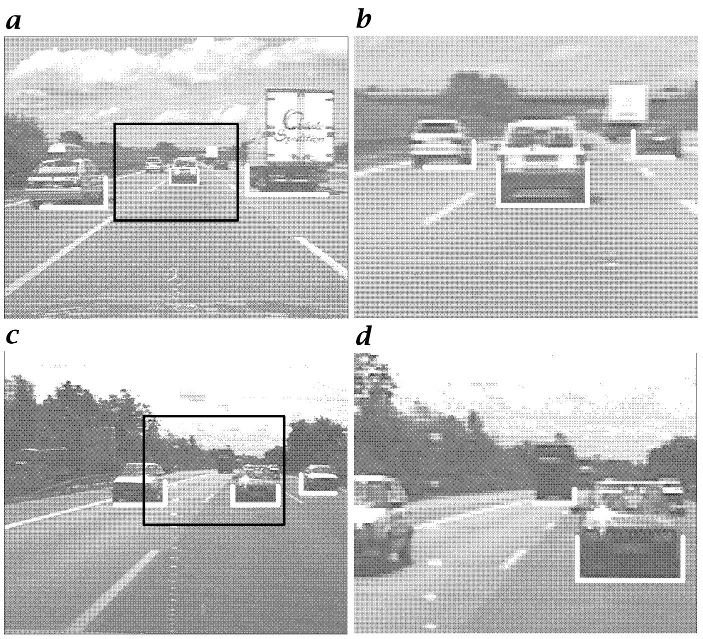
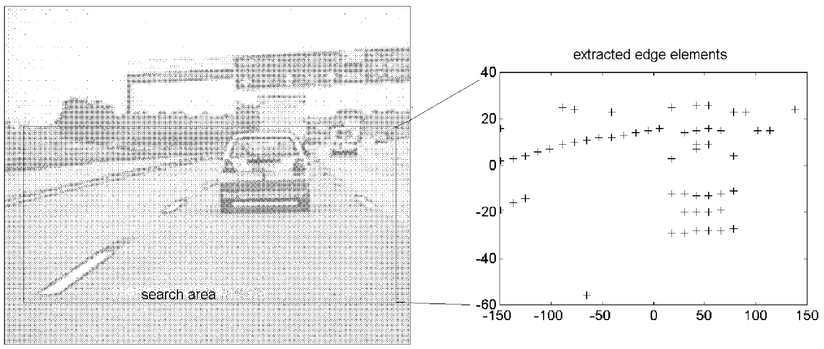


Figure 28.19: *Detection and relative state estimation to multiple objects in parallel. In two different lookahead ranges relative positions and speed components are measured.*

- *Relative state to other vehicles: Obstacle avoidance* is of utmost importance in road vehicle guidance. The nearest vehicles in the lane where the vehicle is driving and the neighboring lanes should be tracked. With bifocal vision, three objects are tracked within an image sequence of each camera (see Fig. 28.19). Because of the occlusion problem in the vehicle's lane, here usually, the object tracked both in the wide angle and in the tele-image is the same; this has to be verified in a separate step. In this case, the same object model may be driven with two sets of data based on different measurement models. The elements of the Jacobian matrices underlying recursive estimation are different in this case. If the two cameras are arranged at a proper distance, even binocular (bifocal) stereo should be possible but has not yet been tried.

  Figure 28.20 explains how objects are detected and recognized. Below the horizon line, horizontal edges are searched for in multiple vertical search paths (Fig. 28.20b, left) with correlation masks of size $m = 5$, $n_1 = 3$ and $n_0 = 1$ (Fig. 28.20b, right). Figure 28.20a, right shows the edge feature positions detected corresponding to the image to the left. The cluster around 50 is likely to be an object; there are several groups of nearly co-linear and parallel features *a*



b

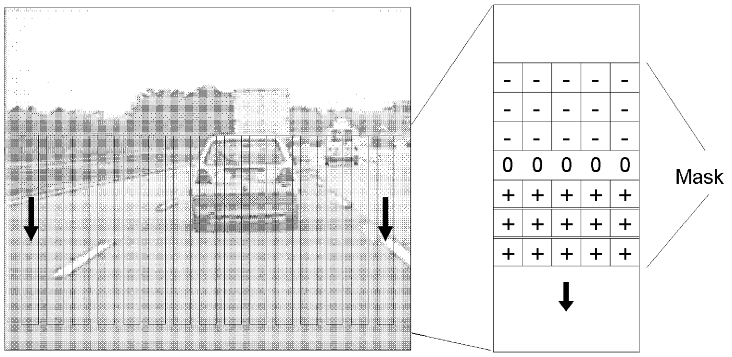


Figure 28.20: *Detection of feature groupings below the horizon line as first step for relative state estimation to multiple objects in parallel:* ***a*** *detection and derivation of measurement data for relative state estimation in near range; localization of regions for finding lateral edges of the object;* ***b*** *search region for the detection of horizontal edge features by the template to the right.*

with proper extension in the lateral direction. The lowest edge is considered to be the line where the object touches the ground.

Next, the lateral boundaries of the object are searched for above the lowest group of features. In each pixel row, the locations of extremal intensity gradients are determined; note that in Fig. 28.21 in some regions the vehicle is darker than the road (bottom part) while it is brighter in others (upper part). The histogram of positions of extremal gradient magnitude counts both (Fig. 28.21). Thus, the width of the vehicle is determined by the difference of the positions of two peaks; the estimated range in conjunction with the mapping parameters is used for conversion from pixels to absolute width.

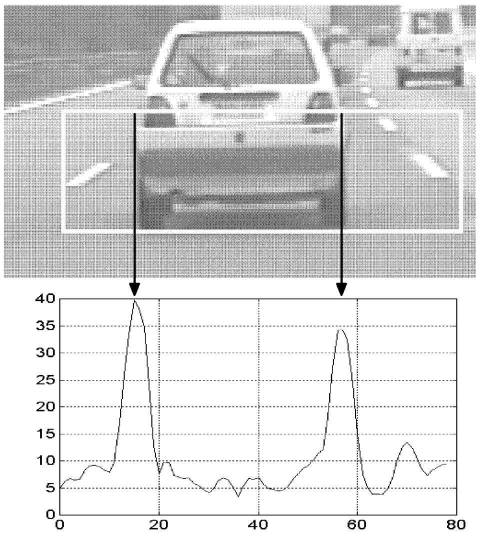


Figure 28.21: *Localization of lateral edges by line-wise edge detection and forming of location histograms (absolute values).*

In 1994, five 25-MHz *transputers* allowed tracking of up to three objects in each image sequence at a rate of 12.5 Hz (80-ms cycle time). In 1995, the transputers had been replaced by *PowerPCs* MPC 601 with an order of magnitude increased computing power. The same process then ran on a single processor at 25 Hz (40-ms cycle time) with improved robustness. For a long-range trip (about 1600 km) to a project meeting in Odense, Denmark in 1995, almost 95 % of the distance was traveled fully automatically, in both *longitudinal and lateral* degrees of freedom. Maximum speed on a free stretch in the northern German plain was 180 km/h.

Because only black-and-white video signals have been evaluated with edge feature extraction algorithms, construction sites with yellow markings on top of the white ones could not be handled. Also, passing vehicles cutting into the vehicle's lane very near in front of the ego-vehicle posed problems because they could not be picked up early enough due to lack of simultaneous field of view, and because *monocular range estimation* took too long to converge to a stable interpretation. For these reasons, the system is now being improved with a wide field of view from two divergently oriented wide angle cameras with a central region of overlap for stereo interpretation; additionally, a high resolution (3-chip) color camera also covers the central part of the stereo field of view. This allows for trinocular stereo and area-based object recognition.

Dual-PentiumPro processors now provide the processing power for tens of thousands of mask evaluations with CRONOS per video cycle and processor.

VaMoRs, the 5-ton van in operation since 1985, which has demonstrated quite a few “firsts” in autonomous road driving, has seen the sequence of microprocessors from Intel 8086, 80×86, via transputers and PowerPCs back to general purpose Intel *Pentium*, PentiumPro and Pentium II. In addition to early high-speed driving on freeways [44], **VaMoRs** has demonstrated its capability of driving on both state and minor unsealed roads at speeds of up to 50 km/h (1992). It is able to recognize hilly terrain and to estimate both vertical and horizontal road curvature.

Recognizing crossroads of unknown width and angular orientation has been demonstrated as well as turning off onto these roads, even with tight curves requiring an initial maneuver to the opposite direction of the curve [39, 45]. These capabilities will also be considerably improved by the new camera arrangement with a wide simultaneous field of view and area-based color image processing.

Performing entire missions based on digital maps is underway [38] and is facilitated now by a GPS-receiver in combination with recently introduced inertial state estimation [34, 39]. The vehicles **VaMoRs** and **VaMP** together have accumulated a record of over 10,000 km of fully autonomous driving on many types of roadways.

In total, seven road and all-terrain vehicles have been equipped with UBM vision systems over the last decade. They all performed both longitudinal and lateral guidance autonomously based on vision [46, 47]. Since 1992 several of these vehicles took part in public Autobahn and high-speed-road traffic.

28.6.2 Autonomous visual navigation of air vehicles

Landing approaches of conventional aircraft. Initially, the feasibility of on-board autonomous control in all six degrees of freedom by dynamic machine vision has been demonstrated for the case of straight-in, unperturbed landing approaches in hardware-in-the-loop simulations [48]. A second effort, including inertial sensing and both wind and gust disturbances, led to the first flight tests in 1991 [49, 50]. Due to safety regulations, the autonomous vision system was not allowed to control the aircraft, a twin turbo-prop Do 128 of about 5-ton weight, near the ground. The human pilot did the flying but the vision system determined all 12 state-components relative to the runway for distances below 900 m from the runway threshold.

The next step was to introduce bifocal vision with one mild and one stronger telelens in connection with the new transputer system in the early 1990s. In 1993, in another set of flight experiments with the

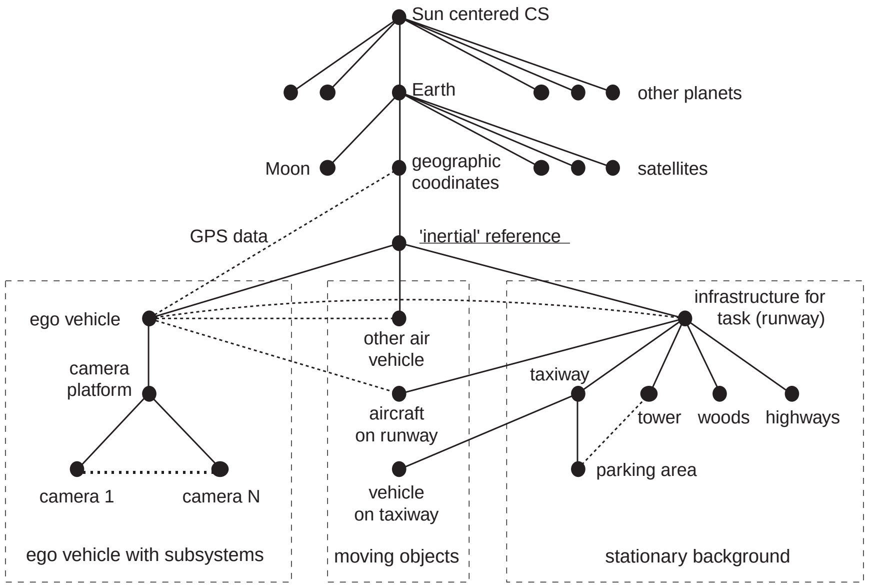


Figure 28.22: *4-D object and scene representation for vision and local navigation; "scene tree" for representation of the situation in an aircraft-landing approach.*

same aircraft at the University of Brunswick, it was proved that visual range could be doubled, essentially, but more computing power would be needed for robust tracking and initialization. The PowerPC satisfied these requirements; it has become possible to detect large obstacles on the runway sufficiently early for safe reactions [30].

Figure 28.22 shows a scene tree for situation assessment in the new, third-generation implementation under development at UBM. It even allows taking lighting conditions by the sun into account for any time at any point on the globe.

Helicopter landmark navigation. The most demanding guidance and control task performed up to now in hardware-in-the-loop real-time simulations is helicopter flight near the ground, including landmark navigation (see Fig. 28.23). The capability of performing a small-scale mission has been shown. Starting at one end of the Brunswick airport, the simulated helicopter flew autonomously by real-time machine vision along a selected sequence of way-points on the airport and in the vicinity. Road forks (Fig. 28.23a), markings on the runway, and the letter H at the destination (Fig. 28.23b) were the landmarks followed. Returning to the airport from the other side and slowing down for landing at the helicopter marker (H) at the other end was demonstrated [34, 35].

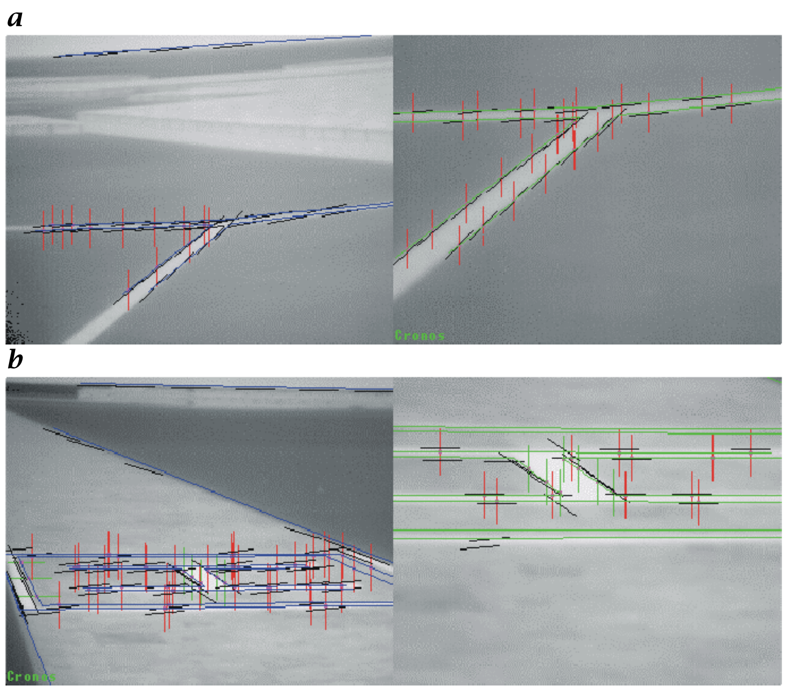


Figure 28.23: *Visual landmark navigation for an autonomous helicopter using a road junction as a way-point for mission performance has been demonstrated in hardware-in-the-loop, real-time simulations for a small mission near the airport of Brunswick [34]:* ***a*** *tracking of "crossing 2": mild- (left) and stronger telelens (right);* ***b*** *tracking of taxiways, frame and helicopter-H during final approach.*

In connection with this demanding task, a complete software package has been developed containing separate inertial and visual state estimation components. Integration of GPS signals and data fusion in the context of mission performance has been achieved.

In addition, provisions have been made to integrate coarse-scale image data from a synthetic aperture imaging radar system under development elsewhere. The combined use of all-weather radar images and high-resolution optical or infrared images is considered an optimal solution for future helicopter guidance systems. The capability of interpreting these data streams by an intelligent on-board computer system will relieve the pilot from a very difficult task in a situation where he is already stressed to the limit.

28.7 Conclusions and outlook

Based on spatiotemporal world models oriented towards physical objects and their motions, intelligent control of feature extraction makes dynamic machine vision possible with the rather limited computing power that is currently available. Special feature extraction algorithms realizing the application of fast “1-D” variants of well-known methods, based on the idea of “receptive fields” for pixel processing, allows efficient real-time operation in closed loop form to exploit prediction error feedback.

Jumping to object hypotheses rather early in the interpretation process and checking several likely hypotheses in parallel over time may be more efficient than trying to follow the bottom-up feature aggregation route (with the danger of combinatorial explosion) too far. Four-dimensional object hypotheses may be checked both in 3-D space and over time; in both dimensions characteristic behaviors may be analyzed and combined, allowing more efficient hypothesis pruning. Computing power is becoming available now that starts several recursive estimation loops for each object in parallel, if necessary.

The 4-D approach to dynamic machine vision developed along the lines laid out by cybernetics and conventional engineering many years ago seems to satisfy all the expectations it shares with “Artificial Intelligence”- and “Neural Net”-approaches. Complex perception and control processes like ground vehicle guidance under diverse conditions and in rather complex scenes have been demonstrated as well as maneuver- and mission-control for air vehicles over a full six degrees of freedom. The representational tools of computer graphics and simulation have been complemented for dealing with the inverse problem of computer vision.

The lack of robustness encountered to date due to black-and-white as-well-as edge-based image understanding can now be complemented by area-based representations including color and texture, both of which are very demanding with respect to processing power.

28.8 References

[1] Thompson, A. M., (1977). The navigation system of the JPL robot. In *Proc. IJCAI*, pp. 749–757.

[2] Moravec, H., (1983). The Stanford cart and the CMU rover. In *Proc. IEEE*, Vol. 71, pp. 872–884.

[3] Bertozzi, M., Broggi, A., Conti, G., and Fascioli, A., (1997). Obstacle and lane detection on ARGO. In *Proc. IEEE—Conf. on Intelligent Transportation Systems, Boston*. IEEE.

[4] Evans, J., (1990). Filtering of pose estimates generated by the RAPiD tracker in applications. In *Proc. Brit. Mach. Vis. Conf.*, pp. 79–84.

[5] Kluge, K., (1997). Performance evaluation of vision-based lane sensing: Some preliminary tools, metrics, and results. In *Proc. IEEE - Conf. on Intelligent Transportation Systems, Boston.*

[6] McLauchlan, P. and Malik, J., (1997). Vision for longitudinal vehicle control. In *Proc. IEEE—Conf. on Intelligent Transportation Systems, Boston.*

[7] Willersinn, D. and Enkelmann, W., (1997). Robust obstacle detection and tracking by motion analysis. In *Proc. IEEE—Conf. on Intelligent Transportation Systems, Boston.*

[8] Zomoter, Z. and Franke, U., (1997). Sensor fusion for improved vision based lane recognition and object tracking with range-finders. In *Proc. IEEE—Conf. on Intelligent Transportation Systems, Boston.*

[9] Bajcsy, R., (1988). Active perception. *Proc. IEEE*, **76**:996–1005.

[10] Aloimonos, J., (1990). Purposive and qualitative active vision. In *10th Int. Conf. on Pattern Recognition*, pp. 346–360. IEEE.

[11] Ballard, D., (1991). Animate vision. *Artificial Intelligence*, **48**:57–86.

[12] Terzopoulos, D. and Metaxas, D., (1992). Tracking nonrigid 3D objects. In *Active Vision*, A. Blake and A. Yuille, eds. Cambridge, MA: MIT-Press.

[13] Schiehlen, J., (1995). *Kameraplattformen für aktiv sehende Fahrzeuge.* PhD thesis, UniBw München, LRT, München.

[14] Dickmanns, E., (1995). Road vehicle eyes for high precision navigation. In *High Precision Navigation*, L. et al., ed., pp. 329–336. Bonn: Dümmler Verlag.

[15] Luenberger, D. G., (1964). Observing the state of a linear system. In *IEEE Trans. Mil. Electronics*, Vol. 8, pp. 290–293.

[16] Meissner, H., (1982). *Steuerung dynamischer Systeme aufgrund bildhafter Informationen.* PhD thesis, UniBwM, LRT, München.

[17] Meissner, H. and Dickmanns, E., (1983). Control of an unstable plant by computer vision. In *Image Sequence Processing and Dynamic Scene Analysis*, T. Huang, ed., pp. 532–548. Berlin: Springer.

[18] Kalman, R. E., (1960). A new approach to linear filtering and prediction problems. *Trans. ASMEE J. Basic Eng.*, **82**:35–45.

[19] Wünsche, H.-J., (1986). Detection and control of mobile robot motion by real-time computer vision. In *Advances in Intelligent Robotics Systems. Proc. SPIE*, N. Marquino, ed., Vol. 727, pp. 100–109.

[20] Wünsche, H.-J., (1988). Bewegungssteuerung durch Rechnersehen. In *Fachberichte Messen, Steuern, Regeln Bd. 10.* Berlin: Springer.

[21] DeCarlo, D. and Metaxas, D., (1996). The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *Proc. IEEE—CVPR, San Francisco, CA*, pp. 231–238.

[22] Schmid, M., (1994). *3-D-Erkennung von Fahrzeugen in Echtzeit aus monokularen Bildfolgen.* PhD thesis, UniBwM, LRT, München.

[23] Thomanek, F., (1996). *Visuelle Erkennung und Zustandsschätzung von mehreren Straßenfahrzeugen zur autonomen Fahrzeugführung.* PhD thesis, UniBwM, LRT.

[24] Schick, J., (1992). *Gleichzeitige Erkennung von Form und Bewegung durch Rechnersehen.* PhD thesis, UniBwM, LRT, München.

[25] Schick, J. and Dickmanns, E., (1991). Simultaneous estimation of 3-D shape and motion of objects by computer vision. In *IEEE Workshop on Visual Motion, Princeton, N.J.*

[26] Kuhnert, K., (1988). *Zur Echtzeit-Bildfolgenanalyse mit Vorwissen.* PhD thesis, UniBwM, LRT, München.

[27] Mysliwetz, B., (1990). *Parallelrechner-basierte Bildfolgen-Interpretation zur autonomen Fahrzeugsteuerung.* PhD thesis, UniBwM, LRT, München.

[28] Dickmanns, E. and Graefe, V., (1988). Dynamic monocular machine vision. Machine vision and applications. *Springer International*, **1**:223–240.

[29] Dickmanns, D., (1997). *Rahmensystem für visuelle Wahrnehmung veränderlicher Szenen durch Computer.* PhD thesis, UniBwM, Informatik, München.

[30] Fürst, S., Werner, S., Dickmanns, D., and Dickmanns, E., (1997). Landmark navigation and autonomous landing approach with obstacle detection for aircraft. In *AeroSense '97, Orlando, FL.*

[31] Mallat, S., (1989). A theory of multiresolution signal decomposition: the wavelet representation. In *IEEE Trans. PAMI*, Vol. 11, pp. 674–693.

[32] Hearn, D. and Baker, M., (1986). *Computer Graphics.* Englewood Cliffs, NJ: Prentice Hall.

[33] Burt, P. J., Hong, T. H., and Rosenfeld, A., (1981). Segmentation and estimation of image region properties through cooperative hierarchical computation. *IEEE Trans. Syst. Man & Cybern*, **11(12)**:802–825.

[34] Werner, S., (1997). *Maschinelle Wahrnehmung für den bord-autonomen automatischen Hubschauberflug.* PhD thesis, UniBwM, LRT, München.

[35] Werner, S., Fürst, S., Dickmanns, D., and Dickmanns, E., (1996). A vision-based multisensor machine perception system for autonomous aircraft landing approach. In *Enhanced and Synthetic Vision AeroSense '96, Orlando, FL.*

[36] Maybeck, P. S., (1979). *Stochastic models, estimation and control*, Vol. 1. New York: Academic Press.

[37] Kailath, T., (1980). *Linear Systems.* Englewood Cliffs, N.J.: Prentice-Hall.

[38] Hock, C., (1994). *Wissensbasierte Fahrzeugführung mit Landmarken für autonome Roboter.* PhD thesis, UniBwM, LRT, München.

[39] Müller, N., (1996). *Autonomes Manövrieren und Navigieren mit einem sehenden Straßenfahrzeug.* PhD thesis, UniBwM, LRT, München.

[40] Dickmanns, E., (1998). Information processing architecture for mission performance of autonomous systems capable of dynamic vision. In *SCI Symposium on 'The Application of Information Technologies (Computer Science) to Mission Systems', Monterey, USA.*

[41] Arbib, M. A. and Hanson, A. R. (eds.), (1987). *Vision, Brain, and Cooperative Computing.* Cambridge MA: MIT Press.

[42] Spillmann, W., (1990). *Visual Perception. The Neurophysiological Foundations.* Boston: Academic Press.

[43] Behringer, R., (1996). *Visuelle Erkennung und Interpretation des Fahrspurverlaufes durch Rechnersehen für ein autonomes Straßenfahrzeug.* PhD thesis, UniBw München, LRT, München.

[44] Dickmanns, E. and Zapp, A., (1987). Autonomous high speed road vehicle guidance by computer vision. In *10th IFAC World Congress Munich, Preprint*, Vol. 4, pp. 232–237.

[45] Dickmanns, E. and Müller, N., (1995). Scene recognition and navigation capabilities for lane changes and turns in vision-based vehicle guidance. In *Control Engineering Practice, 2nd IFAC Conf. on Intelligent Autonomous Vehicles-95, Helsinki.*

[46] Dickmanns, E. D., (1995). Performance improvements for autonomous road vehicles. In *Int. Conf.on Intelligent Autonomous Systems (IAS-4), Karlsruhe.*

[47] UniBw München, LRT, (1996). 20 Jahre ISF. Festschrift.

[48] Eberl, G., (1987). *Automatischer Landeanflug durch Rechnersehen.* PhD thesis, UniBwM, LRT, München.

[49] Dickmanns, E. and Schell, F., (1992). Visual autonomous automatic landing of airplanes. In *AGARD Symp. on Advances in Guidance and Control of Precision Guided Weapons, Ottawa.*

[50] Schell, R., (1992). *Bordautonomer automatischer Landeanflug aufgrund bildhafter und inertialer Meßdatenauswertung.* PhD thesis, UniBw München, LRT, München.

Part III

Scientific Applications

29 Depth-from-Focus for the Measurement of Size Distributions of Small Particles

Peter Geißler[1] and Thomas Scholz[1,2]

[1] Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg, Heidelberg, Germany
[2]now with SAP AG, Walldorf, Germany

29.1 Introduction

Counting and measuring size statistics of free-floating particles is a common problem in technical, biological and other applications. Close-range imaging used to observe small or even microscopic particles shows a small depth of field, typically much smaller than the volume

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

the particles float in. Therefore, the measuring volume is determined by the optics and the image processing method, and not *a priori* known. First a suitable method for the determination of the volume has to be found. This chapter focuses on the use of *depth-from-focus* methods for that problem.

Two methods of depth-from-focus to measure concentrations and size statistics of small particles in an oceanographic and in a biotechnological application are described. Additional electronic material can be found in `/movies/29` on the CD-ROM. Both methods pursue different approaches for the depth reconstruction from single images, taking advantage of the particular properties of the observed particles.

The first method described here has been developed for the measurement of size distributions of oceanic air bubbles. These are entrained to the upper region of the ocean by breaking waves where they contribute to air-sea exchange processes. In order to monitor the bubble populations, an optical sensor based on controlling the measuring volume by depth-from-focus is used. A fast depth-from-focus algorithm is essential to estimate size and position of bubbles for a sufficiently large number of images.

The second technique has been developed in cooperation with the research center of ABB Heidelberg and the University of Hannover. It has been applied for in situ measurements of the concentration of cells in a bioreactor during a fermentation process. In contrast to the first application, which deals with particles of identical shape and brightness but different size, the size of the microorganisms is almost the same for all cells in the same biological state, but their brightness varies. Therefore, the same depth-from-focus method cannot be applied to these images.

Although different depth-from-focus approaches are necessary for the two methods, both have in common that they concentrate on algorithms that need only a single image in order to estimate depth. This is due to the fast movement of the observed particles, by means of which one-image methods benefit from easier handling of image acquisition, because multicamera setups can be avoided.

29.2 Depth-from-focus based measurements

The calculation of size distributions from image sequences of particles requires knowledge of number and size of the observed particles as well as knowledge about the measuring volume. Usually, the measuring volume is determined by means of mechanical delimiters. This gives a fixed and easily handled volume, but also causes flow disturbances that cannot be neglected. To overcome this problem, we use the following approach: The basic idea of depth-from-focus based con-

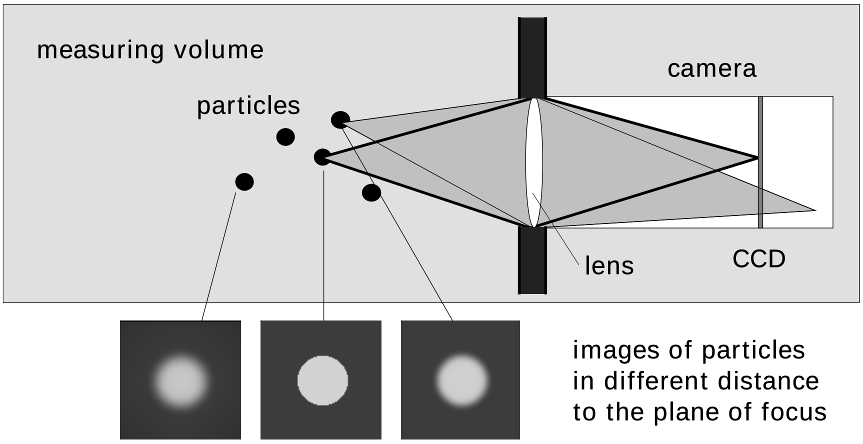


Figure 29.1: *Principle of concentration measurements with depth-from-focus. The 3-D position of each particle is estimated by quantifying its particular defocus.*

centration measurements is the use of a virtual measuring volume. The camera looks freely into the medium, so that the volume defined by its depth of field is much smaller than the volume observed. Both particles inside the measuring volume as well as particles located far outside are imaged. The small depth of field then results in defocused imaging for most particles. The 3-D positions of all observed particles are now determined: The position perpendicular to the optical axis is directly given by the gray-value center of the image of the particle, multiplied by the magnification of the optics, while the position along the optical axis (subsequently denoted by 'z-coordinate') can be inferred from the defocus (Fig. 29.1).

If the 3-D positions are known for a suitable long image sequence, the boundaries of the measuring volume can be determined from these data. If a particle is located too far away from the plane of focus, the width of the point spread function[1] becomes larger than the particle itself. The image is than better described as a 'blurred image of the PSF' than a blurred image of the particle. Therefore no accurate measurement can be done and these particles have to be excluded from the measuring volume. This is done by specifying a minimum value for a measure of the amount of blur. Mechanical delimiters become obsolete to define the measuring volume and are replaced by a *virtual measuring volume* whose boundaries are controlled by the *minimum sharpness criterion*.

[1]The concept of describing defocus by convolution with a point spread function (PSF), has been explained in detail in Chapter 4.7.1.

To obtain the distribution of particle sizes, in addition to the 3-D position the size of each particle has to be measured. The size of blurred particles is not *a priori* given and has to be defined in a suitable manner. This by some means arbitrary measure of size is then combined with the position to correct for the true size of the particle.

It is important to note that the use of depth-from-focus to measure concentrations requires an adequate visualization method to image the particles in a way suitable for the depth-from-focus technique.

29.3 Size distribution of air bubbles

This section describes the instrument for the investigation of oceanic air bubbles with an optical sensor. Natural bubble populations spawn a very large size range of several decades. In order to handle this fact the sensor uses a system of three cameras, each one having a different object-to-image ratio, thus observing different size ranges. A fast image processing algorithm based on depth-from-focus is used to calculate size distributions from the image sequences gained by the instrument.

29.3.1 Physical background

Increasing interest in the understanding of the global climate controlling mechanisms is also drawing more and more interest to investigation of transport processes at the air/water interface of the oceans. Climate relevant gases are exchanged between the two reservoirs, controlled by a thin boundary layer at the very surface. Under rough and windy conditions, bubbles and bubble plumes are generated by wind-induced breaking waves in the upper region of the ocean. During the lifetime of a bubble plume, gases can be transferred without crossing the boundary layer. Thus, gas exchange is enhanced in the presence of bubble plumes. In order to study the underlying processes, it is necessary to measure bubble concentrations and size distributions in the laboratory under controlled conditions as well as in the ocean. Currently available models of bubble mediated air/sea gas exchange depend on reliable estimates of the bubble populations, which are still not available in sufficient numbers, as pointed out by Woolf [1].

29.3.2 Measurement principle

Several different techniques are known for the measurement of submerged air bubbles. Best known are acoustical and optical techniques. Acoustical techniques, which are based on sound attenuation and sound speed variation, have the general problem of being disturbed by sound reflections at the water surface. With the rapid development in video

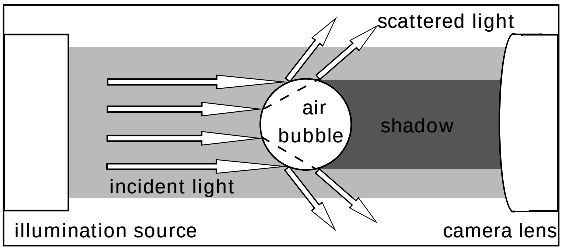


***Figure 29.2:** Light scattering by an air bubble in water.*

technology, it seems appropriate to visualize individual bubbles with CCD cameras, gaining advantage from the possibility to precisely measure bubble sizes and shapes from the images. Due to the large difference in the refractive index of air compared to the surrounding medium, air bubbles show a strong light scattering (Fig. 29.2). In a backlit illumination, most of the incident light is scattered away from the camera lens, and therefore the air bubbles are imaged as dark objects. With no bubbles in the measuring volume, the bright background of the undisturbed illumination is observed.

Natural bubble populations range down to bubble radii in the order of $10\,\mu$m. To image these bubbles with a standard CCD camera having a resolution of 240×512 pixels per field, macro lenses with object-to-image ratios in the order of 1:1 are necessary. These lenses show a small depth of field. Therefore, not all bubbles observed appear in focus, but most of them are blurred due to defocusing. By quantifying the defocus, we are able to correlate it with the distance of the bubble from the plane of best focus. With that information, the 3-D world coordinates of all bubbles observed during the measurement are known. Thus, it is possible to calculate the true measuring volume in situ from the image sequences, without the need to define it by mechanical delimiters using the depth-from-focus principle (Fig. 29.1).

29.3.3 Description of the sensor

In recent years, we have developed several bubble measuring instruments for laboratory measurements, based on the optical technique described here (Geißler and Jähne [2, 3]). An enhanced version of the instrument has been developed for near-shore measurements (Geißler and Jähne [4]). Two major concerns have led to the development of this multicamera sensor. First, the need to perform measurements in the ocean was satisfied by integrating the instrument in a floating buoy. The second major concern was the necessity to record bubbles of all occurring sizes. Natural bubble populations typically show a very large size range, spawning several orders of magnitude from $10\,\mu$m up to more than 1000 μm radius. This goal cannot be achieved with a single camera if a good size resolution is to be maintained even with the smallest bubbles. Therefore a multicamera setup has been chosen with each camera having a different object-to-image ratio.

Illumination. The illumination consists of an array of LEDs, which are pulsed frame synchronized with the cameras. A short pulse duration of not more than 20 μs is necessary to avoid any motion blur. With an observed area of about $1.25 \times 2.5\,\mathrm{mm}^2$ (smallest field of the three cameras) and a camera resolution of 240×512 pixel per field in the RS170 norm, a motion blur of less than one pixel is guaranteed if the bubble velocity stays less than 30 cm/s. This is a very reasonable upper limit for velocity of bubbles beneath the breaking waves.

Camera system. The camera system is the central part of the sensor. It consists of three cameras with different magnifications. A head piece with two beam splitters focuses all cameras at the same spot (Fig. 29.3). This guarantees that the same volume is observed by all cameras. A pair of achromatic lenses forms the optic for each camera. Because all cameras use the same front lens, but have different back lenses, all cameras have the same working distance. Aberrations caused by the beam splitter cubes remain low because the cubes are located in between the achromatic lenses, where the path of rays is nearly parallel. The object to image ratios have been chosen to be 2:1, 1:1 and 1:2, thus resulting in overlapping size ranges. This fact allows consistency checks in the partial spectra obtained from the individual cameras. The 1:1 magnification matches the one-camera instruments, while the two additional magnifications of 1:2 and 2:1 expand the size range towards both ends of the spectrum.

Telecentric optics. As explained here, the position along the optical axis of an air bubble is determined by its blurring caused by the limited depth of field of the optics. As it is not possible to tell if a blurred bubble is located in front or behind the plane of focus, the two possible locations are indistinguishable. Therefore, it is important that this ambiguity does not introduce errors in the size measurement. This is achieved by using a telecentric path of rays (see Volume 1, Chapter 4.6.3), where the aperture stop is located at the back focal point. This causes the principal ray not to intersect the optical axis at the center of the lens, but at the back focal point and therefore to be parallel to the optical axis in object space. As result, the size of an object does not change if it is moving along the optical axis. It is also important to notice that the radius of the blur circle still increases with the distance from the plane of focus, but not longer depends on whether the object

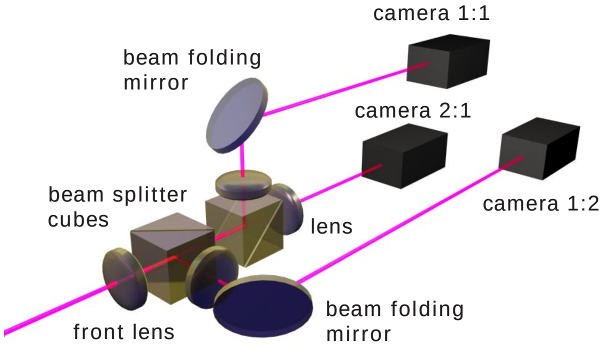


Figure 29.3: *Simplified sketch of the optical system. The two beamsplitters distribute the light to the three cameras. All cameras use the same front lens, but different output lenses—resulting in a set of object-to-image ratios of 1:2, 1:1 and 2:1.*

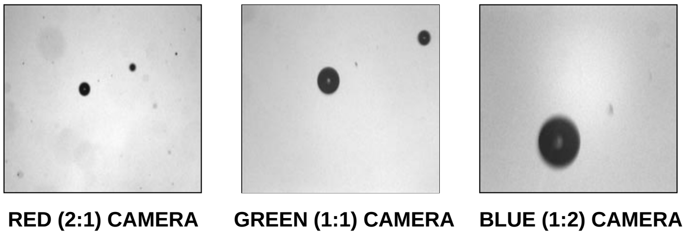


Figure 29.4: *Image of a well-focused air bubble in all three color channels. Because the cameras are focused on the same spot, the same bubble is imaged with three different magnifications.*

is located in front of it or behind. Therefore the use of a telecentric path of rays completely solves the ambiguity problem.

Image acquisition. An RGB color frame grabber is used to simultaneously acquire the images of the three cameras. For later processing, the color images have to be separated into three different images. The image sequences have been stored either digitally or on laser video disc for later processing. An example image is shown in Fig. 29.4

Mechanical setup. To be able to operate underwater, and even at the rough conditions underneath breaking waves, the sensor is protected by two pressure cases, one containing the optics and cameras, the other the light source. The cases are fixed on an aluminum frame that can be

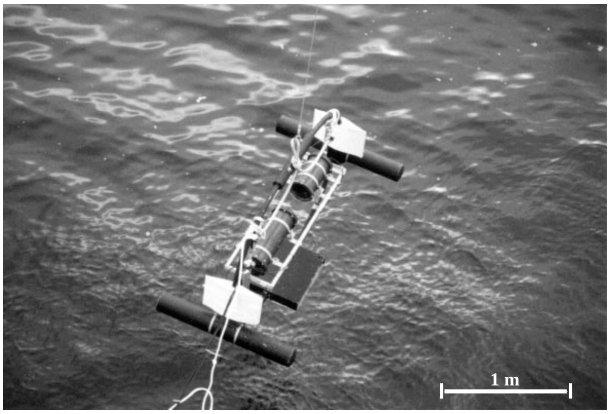


***Figure 29.5:** Deployment of the bubble sensor mounted on the buoy.*

mounted in the wave tank for laboratory measurements. To perform measurements in the ocean, the frame can be mounted on a floating buoy (see Figure 29.5). The buoy is linked by long cables to the image acquisition computer and controlling electronics.

29.3.4 Data processing

Three parameters are necessary to obtain the concentration and size distribution of the bubbles: the total number of bubbles observed during measuring time, their size and the measuring volume. The measuring volume is limited by the depth of field of the optics, which is small compared to the distance between light source and camera system. It can be calculated in an efficient manner by evaluating a sequence of bubble images. This is possible taking advantage of the fact that blurring increases with increasing distance of the object from the well-focused position, and that due to the small depth of field significant blurring occurs in the images. A depth-from-focus algorithm developed by the authors is used to quantify the amount of defocus. This parameter is then used to calculate the bubble's distance from the location of exact focus. Then, we know the exact 3-D position for each individual bubble. This set of position measurements in 3-D world coordinates allows for the calculation of the measuring volume. Most common depth-from-focus techniques need several images from the same scene (Ens and Lawrence [5]), which can hardly be obtained from fast moving objects. We have developed an algorithm that takes into account the known shape of the objects and needs only one image. A region-growing segmentation method is used to determine bubble sizes [3, 6]. Together with the position measurement provided by the depth-from-focus, the true size of the bubbles can be precisely measured even from blurred images. The algorithms used are described in detail in Section 29.3.5. Although the image sequences have been stored for safety reasons, the algorithm is fast enough to perform on-line processing on a suitable Pentium-based system. Real-time processing should become possible in the future with the use of MMX instructions.

29.3.5 Image processing and depth-from-focus

Position ambiguity. As already mentioned in the foregoing, it is not possible to distinguish whether a bubble is located in front of or behind the plane of focus. The calculation of the true size of a bubble from its blurred size and the amount of blur does not result in an ambiguity of the radius measurement, because a telecentric ray path has been used. It is important to note that the position ambiguity cannot be resolved, but handled with the appropriate optics.

The processing of the images to determine the size distribution consists of two main steps: segmentation and depth-from-focus. Before these steps are performed, a brightness normalization is applied to the images to eliminate inhomogeneities and brightness variations of the illumination.

Brightness normalization. A good understanding of the image formation process and a normalization of the gray values, which is independent of brightness fluctuations and inhomogeneities, is an essential requirement for the depth-from-focus technique. The normalization is done by applying a linear illumination model assuming that the measured gray values $g(\boldsymbol{x})$ are further proportional to the irradiance $I(\boldsymbol{x})$. Then

$$g(\boldsymbol{x}) = a(\boldsymbol{x})I(\boldsymbol{x}) + b(\boldsymbol{x}) \tag{29.1}$$

The unknown quantities $b(\boldsymbol{x})$ and $a(\boldsymbol{x})$ are obtained by taking a background image $g_b(\boldsymbol{x})$ with illumination switched off ($I(\boldsymbol{x}) = 0$) and a zero image $g_z(\boldsymbol{x})$, in which no bubbles are present ($I(\boldsymbol{x}) = I_0(\boldsymbol{x})$).

If we describe the objects by their light absorbing coefficient $\tau(\boldsymbol{x})$ in the object plane (capitals denote object plane coordinates and small letters denote image plane coordinates) their image is given by :

$$I(\boldsymbol{x}) = v(\boldsymbol{x})\left[1 - \tau\left(\frac{\boldsymbol{x}}{V_g(z)}\right) * PSF_z(\boldsymbol{x})\right] \tag{29.2}$$

with $v(\boldsymbol{x})$ describing vignetting and $V_g z$ being the magnification. Then, the linear inhomogeneous point operation

$$n(\boldsymbol{x}) = \frac{g_z(\boldsymbol{x}) - g(\boldsymbol{x})}{g_z(\boldsymbol{x}) - g_b(\boldsymbol{x})} = 1 - \frac{I(\boldsymbol{x})}{I_0(\boldsymbol{x})} = \tau\left(\frac{\boldsymbol{x}}{V_g(z)}\right) * PSF_z(\boldsymbol{x}) \tag{29.3}$$

results in a normalized gray value n in the range of 0 to 1.

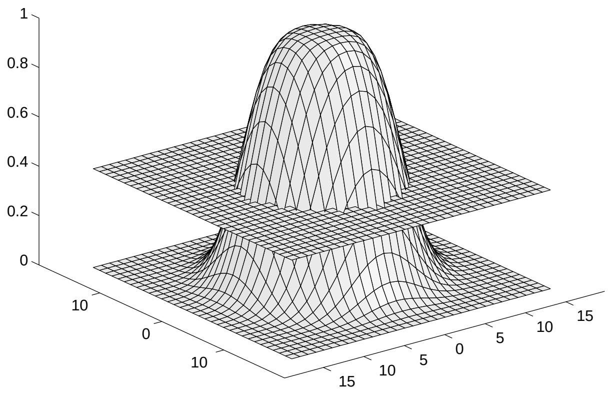


Figure 29.6: *Definition of the $1/q$-area as the size of blurred objects. As an example, the image shows a blurred object and its boundary given by the intersection with the $1/q = 0.4$ plane.*

Segmentation. The image processing step of the segmentation distinguishes objects from the background and calculates their apparent (blurred) size. After the depth-from-focus calculation has been performed, the apparent size is corrected to the true particle size. Because blur causes the originally step edges of the objects to become flat, the boundary of a blurred object is not *a priori* well defined. Therefore we define the boundary to be at these locations where the gray value has decreased to the $1/q$-th of the maximum gray value (Fig. 29.6).

The method used to segment the bubbles is a two-step approach that combines a presegmentation step with a fast region growing algorithm. Bubbles within the largest possible size of the measuring volume show a plateau with a gray value of 1 in the normalized image. At the very border of that volume, the plateau shrinks to a single point. Beyond this maximum distance from the focal plane, the width of the PSF exceeds the size of the (well-focused) image of the bubble. For that reason it is no longer possible to calculate size and depth from the blurred image and, therefore, it is not necessary for the presegmentation to detect those bubbles. Because all bubbles that may be within the measuring volume have to show a maximum gray value of about 1 and the background has been made uniform by the normalization, presegmentation can be carried out by a global threshold. It is important to note that the value of the threshold does not affect the result of the segmentation, as it is guaranteed that all bubbles within the measuring volume are found as long as the threshold is within a sensible range, for example, 0.2 to 0.8.

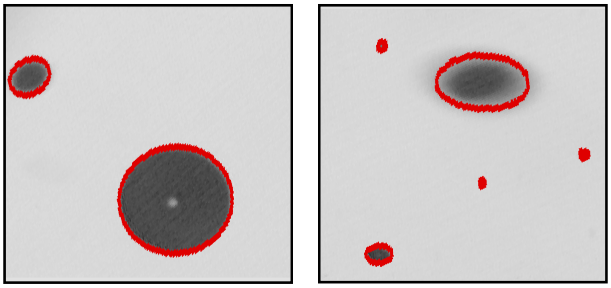


Figure 29.7: *Final segmentation result of two images. The gray lines indicate the boundary of the particles (obtained with $1/q = 1/e$).*

The exact boundary of a bubble is found by the second step, the region-growing algorithm. This algorithm is a modification of a region growing method developed by Hering et al. [6] and shall be briefly described here. The initial step of a region growing segmentation is the detection of *seeding points* as starting locations for the growing. With our algorithm, seeding points are defined as the location of the gray-value maximum of each object. The image is smoothed by a small binomial filter to reduce noise and therefore avoid mislocating the maximum due to random noise peaks. The region growing phase starts with each seed defining different objects, which consist of this single pixel each. Pixels are then added to the objects if their gray value is larger as $1/q$ times the gray value of the initial seeding point and if they are 8-neighbors of the current boundary line of the object. The growing phase stops if no new object pixels can be found. The region growing procedure causes the objects to be connected and to have the correct size, regardless of their size in the starting image provided by the thresholding. Figure 29.7 shows the final result of the segmentation for several bubbles.

29.3.6 Depth-from-focus

A usual approach for depth-from-focus is to calculate a measure of blur at each image point. Thus, a depth-map of the image can be calculated which contains the distance from the plane of focus for each pixel. This is only possible if more than one image of the same scene is available, due to the impossibility of distinguishing between PSF and object function from the gray value image (see Volume 2, Chapter 20). Our approach differs from that of calculating a depth-map in that it performs the object detection first and then does an object-oriented depth-from-focus, measuring the amount of blur of complete objects. This allows for a fast depth determination, suitable for the evaluation of long image sequences.

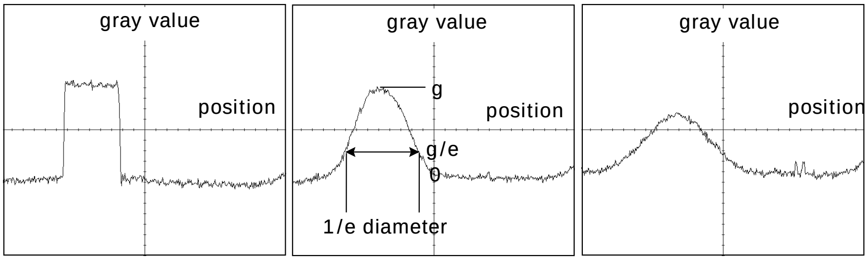


Figure 29.8: *Radial cross section through the gray-value distribution of bubbles at different distance from the plane of focus. The distances increases from left to right.*

A good integral measure of the blur of a particle is the mean gray value g_m on the segmented area. With increasing blurring, the edges of the particles become less steep and, therefore, the mean gray value decreases (Figs. 29.8 and 29.11).

The fact that the shape of the point spread function is independent of the distance z allows the PSF to be expressed in terms of a general shape function $B(\boldsymbol{x})$ and a scaling factor $V_p(z)$:

$$PSF_z(\boldsymbol{x}) = kB\left(\frac{1}{V_p(z)}\boldsymbol{x}\right) \tag{29.4}$$

with the normalization factor $k^{-1} = \int d\boldsymbol{x} B(\boldsymbol{x})$. At the plane of focus ($z = 0$) V_p is zero, resulting in a delta function for the PSF. This can be used to analyze the behavior of g_m in more detail.

All bubbles with the same ratio between the radius r' and the size V_p of the point spread function have the same mean gray value g_m, because their images differ only in size and are of the same shape. Thereby, $r' = V_g(z)r$ is the radius of the well focused object of radius r on the image plane. Thus,

$$\frac{V_p(z)}{V_g(z)r} = \text{const} \Leftrightarrow g_m = \text{const} \tag{29.5}$$

Denoting the constant in Eq. (29.5) by $\gamma(g_m)$ and resolving the mean gray value is given by $g_m(z,r) = \gamma^{-1}(V_p(z)/r)$ and, therefore,

$$g_m(z,r) = g_m\left(\frac{V_p(z)}{V_g(z)r}\right) \tag{29.6}$$

If we use a telecentric path of rays the magnification V_g becomes independent from z, and with the use of $q = 1/2$ for segmentation, the 1/q-area represents the true size of the particles. Furthermore, $V_p(z)$

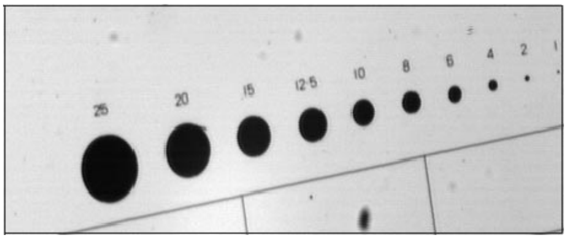


Figure 29.9: *Partial view of the calibration target.*

is then a symmetric and linear function of z. Therefore, g_m depends only on the normalized distance z/r:

$$g_m(z,r) = \gamma^{-1}\left(\frac{a|z|}{r}\right) = g_m\left(\frac{|z|}{r}\right) \tag{29.7}$$

29.3.7 Calibration

Calculating bubble size distributions from the image sequences with the depth-from-focus approach requires that the instrument is carefully calibrated with a focus series of calibration targets of known size. Patterson reticules were used as calibration targets. This standard target for microscopic calibration consists of a series of black circles in a diameter range of from 18 μm up to 450 μm on a glass plate (Fig. 29.9). Because Patterson globes are not absolutely accurate, the true size of each circle has to be measured independently, for example, using a calibrated microscope. A black circle is a very good approximation of the focused image of a bubble, since with the optical setup used in the experiments more than 99.6 % of the incident light is scattered away from the receiver. Nevertheless, the bright dot that appears in the center of well-focused bubbles can be easily removed by applying a median filter to the normalized image. In order to perform the calibration, an optical bench is mounted on top of the device. The calibration target is then fixed to the bench with a precision positioning table.

Depth series centered at the focal plane are taken with a step size of 1 mm. Figure 29.10 shows the radii measured from different circles of the Patterson target. Within the measuring volume, the difference between the measured and the true radius is the order of 10 to 15 μm, which is about the size of one pixel. The variation of the mean gray value with increasing depth is shown in Fig. 29.11. A linear model $g(z,r) = g_0 - \alpha(r)|z|$ fits well to the data. Because a small axis offset and slight tilt of the target can often not be avoided, the axis offset for each circle is corrected by finding the center of symmetry in its depth series.

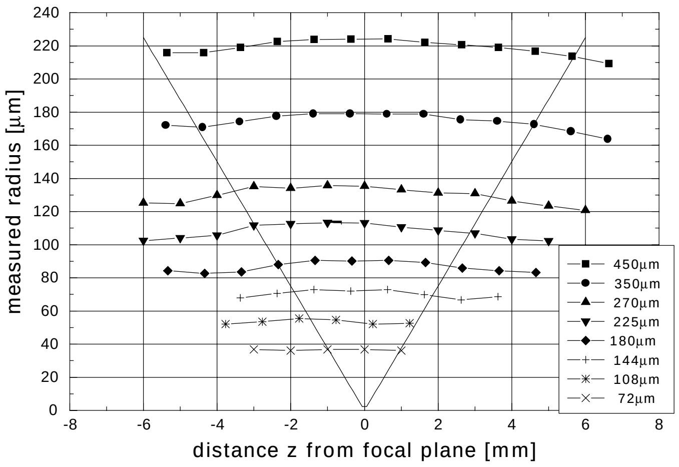


Figure 29.10: *Independence of the size of blurred bubbles with the distance from the plane of focus. The thin lines indicate the maximum size of the virtual measuring volume.*

29.3.8 Calculation of particle concentration

Determination of the measuring volume. The decrease of the mean gray value with increasing distance from the plane of focus can now be used to define the measuring volume by a lower limit for g_m. Only bubbles with mean gray values above this limit are taken into account for the calculation of the size distribution. Thus the linear dependence of g_m on the normalized distance

$$g(z,r) = g_0 - \alpha(r)|z| = g_0 - \alpha_0 \frac{|z|}{r} \tag{29.8}$$

gives the volume boundary:

$$z_{max} = \frac{1}{\alpha(r)}(g_0 - g_{min}) = \frac{r}{\alpha_0}(g_0 - g_{min}) \tag{29.9}$$

The measuring volume is then given by

$$V(r) = 2z_{max}(r)A_0 \tag{29.10}$$

with A_0 being the area imaged by the CCD at the plane of focus. The depth of the volume is controlled by the parameter g_{min}. It is important to note that the volume depends on particle size and increases with larger particles, because a large particle becomes less blurred compared to a smaller particle if its image is convoluted with the same PSF.

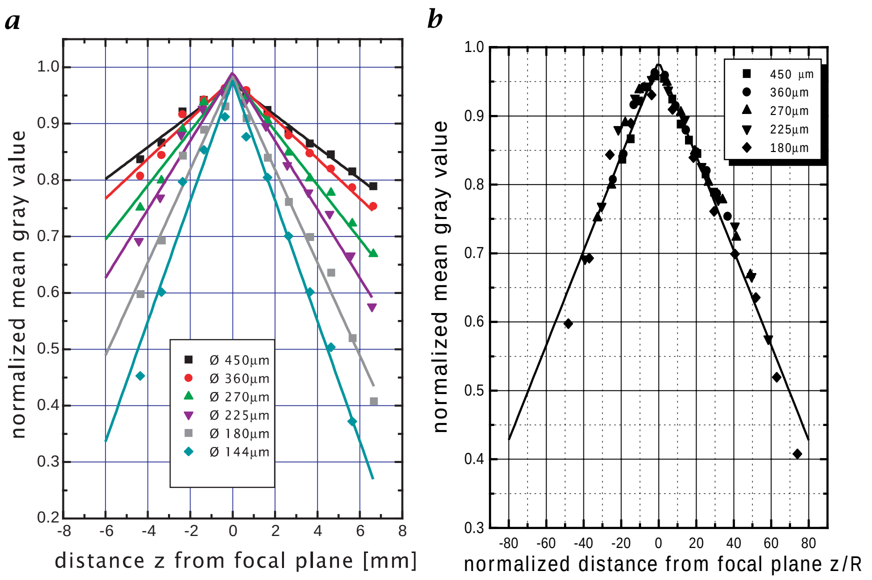


Figure 29.11: ***a*** *mean gray value calculated for different Patterson globes;* ***b*** *mean gray value versus normalized distance z/R. This validates the fact that g_m depends only on z/R.*

Calculation of true particle size. As mentioned here, with a telecentric path of rays the true particle size can be easily obtained from the segmentation of the images. Due to the symmetry between particles located at the same distance, but in front or behind the focal plane, the intrinsic ambivalence does not cause an ambivalence in the depth or size measurement and can be ignored completely. The situation is different with standard optics where the aperture stop is not located at the back focal plane of the lens system. Then, V_g depends on z and the segmented size does not necessarily meet the true size. The fact that there is a unique relation for the true radius r and the depth z of an object to the two measurable parameters, g_m and the segmented radius r_q can be used to solve the problem [7]. The relation between the four parameters is obtained from the calibration depth series. The values of the output parameters (r, z) are mapped on a regular grid in the input parameter set (r_q, g_m) and used as a fast look-up table to perform the calculation.

Size distributions. Segmentation and depth-from-focus result in the knowledge of position and size of the particles observed in an image. The data from a suitable long image sequence is needed to calculate size distributions. The result of the segmentation of an image sequence is shown in Fig. 29.12.

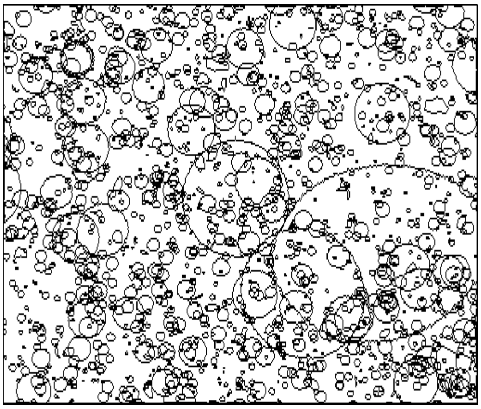


***Figure 29.12:** Result of the segmentation of an image sequence. The dark gray lines indicate the boundary lines of the bubbles.*

Bubble size distributions were calculated from the number $N(r, dr)$ of bubbles found in the radius interval $[r, r + dr]$ by

$$\psi(r, dr) = \frac{N(r, dr)}{N_I dr V(r)} \tag{29.11}$$

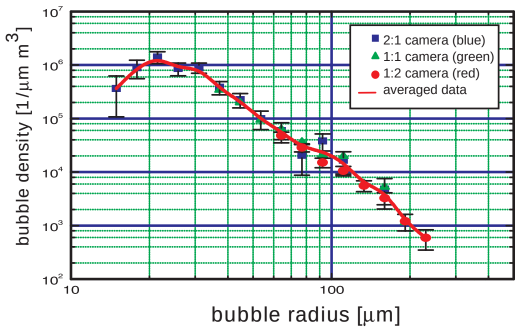
where N_I is the total number of images.

As an example, Figure 29.13a shows some sea water bubble size distributions measured in the large wave tank of the Scripps Inst of Oceanography. These measurements have been described in greater detail in Geißler and Jähne [4].

29.3.9 Results and discussion

The sensor proved its ability to operate both in the laboratory as well as in marine environments. In laboratory studies, the system has been used to investigate bubble populations under artificial breaking waves. The experiments have been performed in the large wave tank at Scripps Institution of Oceanography, San Diego. Figure 29.13a shows the size spectra measured by the three cameras. Towards the lower bubble sizes, the different size ranges covered by the three cameras can be observed. The camera looking at the smallest sizes is able to detect the lower cutoff radius. In addition, comparative experiments using an acoustical technique have been performed under the same conditions by E. Terrill from Scripps institution of oceanography (see Melville et al. [8]). The good agreement between the two methods is shown in Fig. 29.13b. Mounted on the buoy, the sensor has been deployed from the pier of Scripps Institution, both in the near-shore surf zone as well as during a storm event in December 1996. The system behaved well even in rough conditions.

a



b

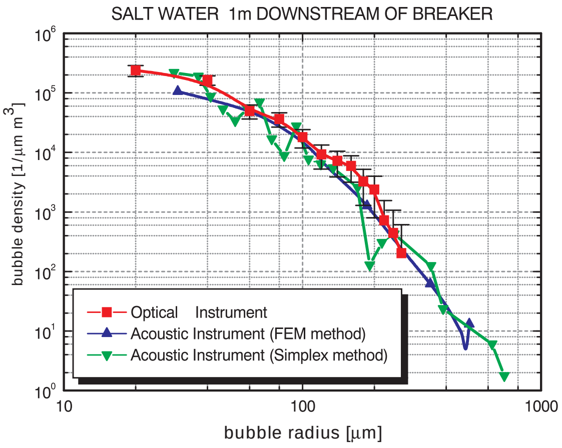


Figure 29.13: ***a*** *Size spectra measured with the three cameras. Due to the different measuring volumes, the partial spectra do not match in absolute number.* ***b*** *Comparison of bubble spectra measured by the optical and by an acoustic method.*

29.4 In situ microscopy

This second application of depth-from-focus based concentration measurement has been developed in order to monitor cell concentration and living biomass in bioreactors. Fermenters are widely used in the biotechnical industry to obtain products by the enzymatic activity of microorganisms. In many of these cultivations, the cell cultures require precise process control, homogeneity in the bioreactor, and high sterility. One of the most difficult assignments is to measure the cell concentration with an appropriate instrument under these conditions.

A depth-from-focus sensor using image processing to measure the cell concentration has significant advantages over other techniques [9]. The defocus of the cell is used to determine a virtual measuring volume

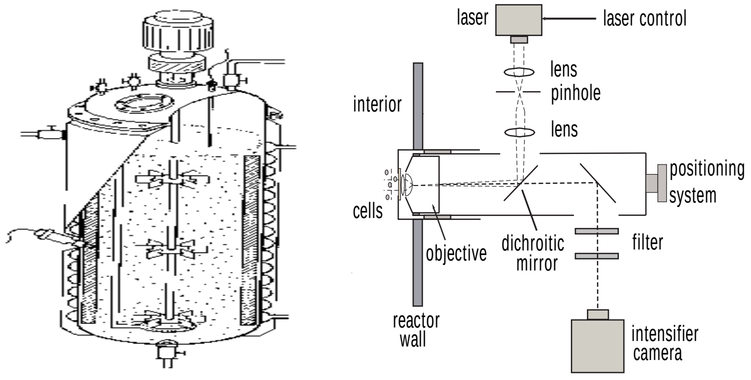


Figure 29.14: *left: Sketch of a typical bioreactor; right: components of the ISM probe.*

from which absolute concentration can be inferred. This noncontact approach avoids all the disadvantages caused by mechanical volume delimiters, bypasses, or sampling. Living cells can be separated from dead cells and other objects such as bubbles or particles by using the fluorescence of the NAD(P)H. The NAD(P)H is an intermediate protein of the metabolic chain and is, therefore, present only in living cells. To realize this concept with automatic image processing and with a steam sterilizable optical setup, a cooperation between Asea Brown Boveri Corporate Research Center, Heidelberg, the Institute of Chemical Engineering, University of Hannover, and the Institute of Environmental Physics, University of Heidelberg, was established.

29.4.1 The in situ microscope

Images of the cell suspension are made by means of a microscope immerged in the bioreactor, the *in situ microscope* (ISM). The in situ depth-from-focus probe is housed in a stainless steel tubing, which was inserted into a port of the bioreactor (see Fig. 29.14b). A microscope objective with a magnification of 100 and a numerical aperture of 0.95 was placed in front of the probe, separated from the interior of the reactor by a quartz slide cover. The cells were rapidly intermixed by a stirrer in the reactor. The illumination was made through the objective with a nitrogen laser at a wavelength of 337 nm and a pulse length of 3 ns. The laser beam is coupled into the imaging light path by a dichroitic mirror. The short illumination time is necessary in order to avoid motion blur due to the fast movement of the cells in combination with the small field of view. The NAD(P)H molecules in the cells absorb the laser wavelength and emit fluorescent light in the visible range at 460 nm. Because of the weak fluorescence of the cells, a light

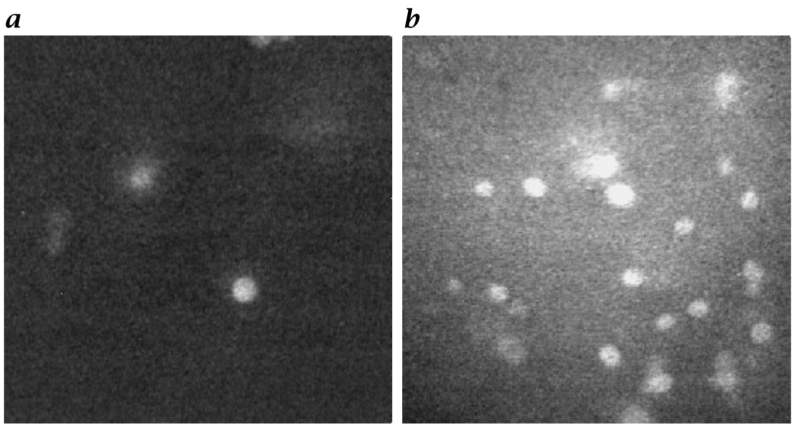


Figure 29.15: *Typical images of cell concentrations of* ***a*** 5×10^5 *cells/ml and* ***b*** 5×10^8 *cells/ml.*

amplifying camera is necessary. Because NAD(P)H is an intermediate product of the metabolism of the cells, only living cells are imaged and errors in the determination of cell concentration caused by dead cells are avoided. The cells (average diameter 5–6 μm) are imaged as circular objects with an almost Gaussian shape (Fig. 29.15). The ISM is described in detail in Suhr et al. [9], see also Schneider [10] and Scholz [11].

29.4.2 Depth-from-focus algorithm

Image preprocessing. Due to the necessity of using a light amplifying camera, the signal-to-noise ratio (SNR) of the images is very poor. On the other hand, noise suppression is essential as the depth-from-focus method makes extensive use of bandpass filters, which tend to be noise sensitive. Therefore, noise suppression is the main step of image preprocessing. As the sharpness of the cells is our indicator for depth, it is not possible to use common smoothing filters to suppress noise. Thus, a suitable adaptive smoothing filter was developed.

Adaptive smoothing. The goal of adaptive smoothing is to preserve the sharpness of objects in the image. To achieve this aim, it is necessary to smooth only along but not across the lines of constant gray values. For that it is sufficient to know the direction of smallest variance of the gray values at each pixel. In order to obtain this direction, the local orientation is calculated from the local variance in different directions (compare Volume 2, Section 10.6). The local variance $\mathrm{Var}_\phi[g(\boldsymbol{x})]$

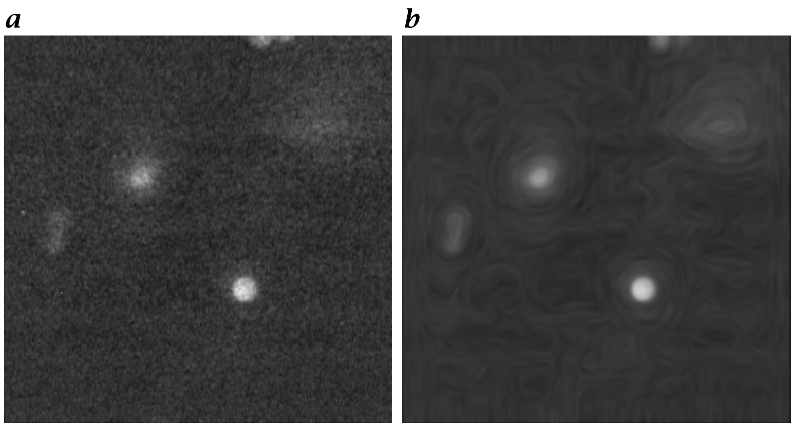


Figure 29.16: ***a*** *Original image.* ***b*** *Result of adaptive smoothing.*

of the gray values $g(\boldsymbol{x})$ in the direction ϕ is given by

$$\begin{aligned} &\mathrm{Var}_\phi[g(\boldsymbol{x})] = E_\phi[(g(\boldsymbol{x}) - E_\phi(g(\boldsymbol{x}))^2] \\ \text{with} \quad &E_\phi[g(\boldsymbol{x})] = \int B_\phi(\lambda)(g(\boldsymbol{x} + \lambda\boldsymbol{e}_\phi)\,\mathrm{d}\lambda \end{aligned} \tag{29.12}$$

as the weighted average of g in ϕ direction, $\boldsymbol{e}_\phi$ is the unit vector, and $B(\lambda)$ is a normalized weighting function. The local orientation is then obtained from the variances in different directions ϕ_i by the vector-addition technique introduced by Knutsson and Granlund [12]. It can be shown that the use of four different directions with an angle of 45° to each other results in a suitable accuracy of the calculation of the orientation direction ϕ. The angle of the local orientation is then given by

$$\phi = \frac{1}{2}\mathrm{arc}\left(\frac{\mathrm{Var}_{\phi_0}[\rho(\boldsymbol{x})] - \mathrm{Var}_{\phi_2}[\rho(\boldsymbol{x})]}{\mathrm{Var}_{\phi_1}[\rho(\boldsymbol{x})] - \mathrm{Var}_{\phi_3}[\rho(\boldsymbol{x})]}\right), \quad \phi_i = i\frac{\pi}{4} \tag{29.13}$$

With this information, it is possible to smooth along the contour lines of the objects in the image. Thus, the course of a contour line through a pixel is evaluated for n pixels in both directions, considering an interpolation between the grid points and an amount of robustness to prevent breakout of the filter at departing values. In Fig. 29.16, an example image smoothed, using the described method is shown.

Segmentation and separation. For image segmentation, the algorithm already described in Section 29.3.5 has been used. Because the depth-from-focus method needs to evaluate each cell separately from the others to get its distance from the focal plane, a separation step follows segmentation. The borderline of each cell in the label image is

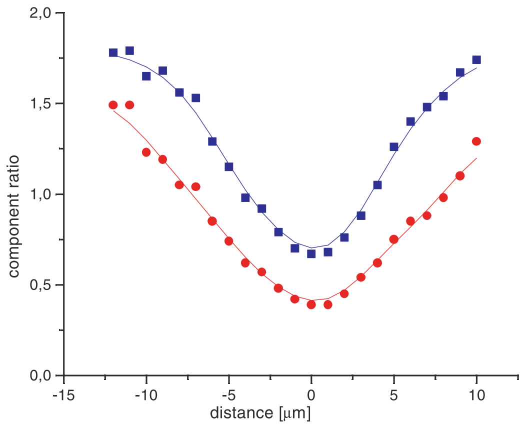


Figure 29.17: *The first two ratios of the feature vector components.*

evaluated and a plane fit through the gray values along the border line is calculated. After that, each cell is embedded in an image and slightly smoothed between fitted plane and edge.

Depth-from-focus on the Laplacian pyramid. As already discussed in Volume 2, Chapter 20, there is no general solution for the depth-from-focus problem. Knowledge about the observed objects and the optical setup is a necessary requirement to get the desired information. In our case, this knowledge is given by the fact that the used yeast cells are objects of approximately similar size and shape and that an identical optical setup with particles similar to the cells was taken to get the calibration series. To get the information about the distance between the cells and the plane of focus, a criterion is required from which the desired depth could be evaluated. A bandpass decomposition can be used to detect the suppression of high spatial frequencies with increasing blur [11]. This approach and its realization with a Laplacian pyramid has been introduced in Volume 2, Chapter 20; the Laplacian pyramid is discussed in Volume 2, Section 4.4.3.

For each cell found by a previous segmentation step, a feature vector is computed by summing the squares of each level of the Laplacian pyramid. It is important to mention that the pyramidal feature vector is calculated for each cell separately to avoid distortions by other cells. The feature vector is, therefore, defined by its components

$$|L^{(k)}|^2 = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (L_{i,j}^{(k)})^2 \tag{29.14}$$

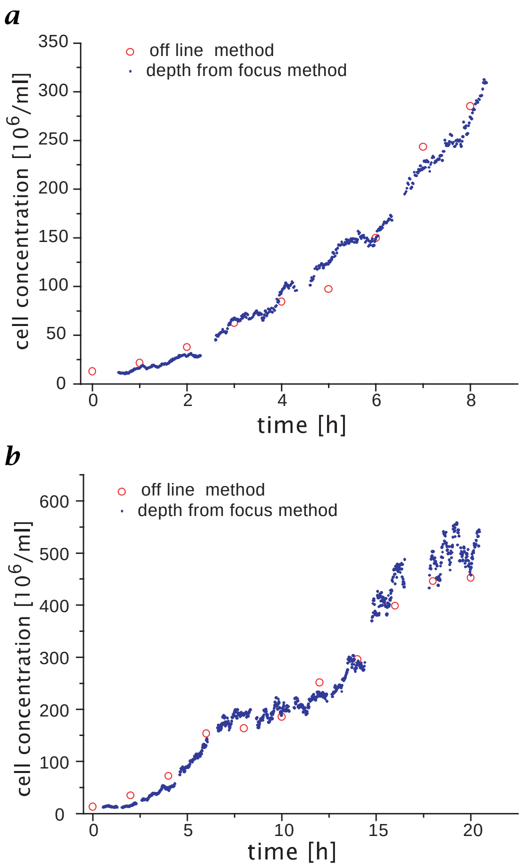


Figure 29.18: ***a*** *Cell concentrations during a fed-batch fermentation , where nutrient solution is added constantly, and during a batch fermentation;* ***b*** *with the only ingestion at the start of the fermentation.*

whereby $L^{(k)}$ denotes the kth level of the Laplacian pyramid. The ratios of the components of the feature vector

$$O_{i,j} = \tan \phi_{i,j} = \frac{|L^{(j)}|^2}{|L^{(i)}|^2} \qquad i, j \in [0, 1, ..., n] \qquad \forall i \neq j \qquad (29.15)$$

are independent of the object brightness and can be used as the final defocus measure.

To get the correct distance of a cell, we have to compare the ratios of the components of the feature vector with the ones of a calibration series using a standard particle. Figure 29.17 shows the ratios of the second and the third component to the first of such a series, determined for a calibration sequence of latex particles with 4.5 μm diameter, which are very similar to yeast cells in shape and size. A precise translation stage was used to move latex particles embedded in gel in z-direction in steps of 0.5 μm. The identical optical setup as used for the imaging of the cells was used to obtain this calibration series. It could be seen that the ratios are almost symmetric around the plane of focus and that the curves could be well approximated by a Gaussian. It is, of course, not possible to differentiate between a particle in front of or behind the plane of focus with this method.

29.4.3 Cell concentrations during fermentation

The described method to get depth-from-focus was tested successfully with some standard series. Furthermore, during a 24-h fermentation process, about 25,000 images were stored with a laser video recorder and offline probes were taken. At the beginning of the cultivation, an image was taken every 2 s, later this interval increased with the concentration of the cells up to 10 s. Figure 29.18 shows the concentration curve of the yeast cell cultivation in the course of two fermentations. It can be seen that the results of the depth-from-focus technique correspond well to those of an offline measurement, taken for control purposes. During fermentation, a slight decrease in cell size with time could be observed. This drift in size causes an error in depth recovery and, therefore, in concentration measurements. However, this error can be compensated by a correction based on an interpolation of the slope of the volume-concentration profile of the series [11].

29.5 Acknowledgments

The authors would like to express their thanks to E. Terrill and W. K. Melville from Scripps Institution of Oceanography for the possibility to use of the wave tank and for providing acoustic datasets. Financial support for this research by the European Community (E.C.) (Large Installation Program) is gratefully acknowledged. Concerning the in situ microscopy project, we would like to thank the ABB Corporate Research Center Heidelberg for financial support.

29.6 References

[1] Woolf, D. K., (1993). Bubbles and the air-sea transfer velocity of gases. *Atmosphere—Ocean*, **31**:517 – 540.

[2] Geißler, P. and Jähne, B., (1995). Measurements of bubble size distributions with an optical technique based on depth from focus. In *Selected Papers from the Third International Symposium on Air Water Gas Transfer, Heidelberg*, B. Jähne and E. C. Monahan, eds., pp. 351 – 362. Hanau: AEON.

[3] Geißler, P. and Jähne, B., (1996). A 3D-sensor for the measurement of particle concentration from image sequences. In *Proc. of the 18th ISPRS Congress, Vienna.* In *Int'l Arch. of Photog. and Rem. Sens.,* **31**, Part B5.

[4] Geißler, P. and Jähne, B., (1997). Laboratory and inshore measurements of bubble size distributions. In *Sea Surface Sound '97—Proceedings of the Fourth International Conference on Natural Physical Processes Associated with Sea Surface Sound*, T. G. Leighton, ed., pp. 147–154. Chilworth Manor, Hampshire, UK: CBC Print and Media Resources.

[5] Ens, J. and Lawrence, P., (1994). An investigation of methods for determining depth from focus. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **15**(2).

[6] Hering, F., Wierzimok, D., and Jähne, B., (1995). Particle tracking in space time sequences. In *6th Int'l Conf. on Computer Analysis of Images and Patterns, Prague, Lecture Notes in Computer Science* Vol. 970. New York: Springer.

[7] Geißler, P. and Jähne, B., (1995). One-image depth from focus for concentration measurements. In *Proc. of ISPRS Intercommission Workshop 'From Pixels to Sequences', Zurich,* In *Int'l Arch. of Photog. and Rem. Sens.,* **30**, Part 5W1.

[8] Melville, W. K., Terrill, E., and Veron, F., (1997). Bubbles and turbulence under breaking waves. In *Sea Surface Sound '97—Proceedings of the Fourth International Conference on Natural Physical Processes Associated with Sea Surface Sound*, T. G. Leighton, ed., pp. 135–146. Chilworth Manor, Hampshire, UK: CBC Print and Media Resources.

[9] Suhr, H., Wehnert, G., Schneider, K., Bittner, C., Scholz, T., Geißler, P., Jähne, B., and Scheper, T., (1995). In-situ microscopy for on-line characterization of cell-populations in bioreactors, including concentration measurements by depth from focus. *Biotechnology and Bioengineering*, **47**:106–116.

[10] Schneider, K., (1994). *In Situ Mikroskopie, eine neue Methode zur Bestimmung der Zellkonzentration während Fermentationsprozessen.* Dissertation, Inst. of Chemical Engineering, University of Hannover.

[11] Scholz, T., (1995). *Ein Depth from Focus-Verfahren zur On-Line Bestimmung der Zellkonzentration bei Fermentationsprozessen.* Dissertation, Fakultät für Physik und Astronomie, Universität Heidelberg.

[12] Knutsson, H. and Granlund, G. H., (1983). Texture analysis using two-dimensional quadrature filters. In *IEEE Workshop Comp. Arch. Patt. Anal. Dat. Base Man., Pasadena, CA.*

30 Fluorescence Imaging of Air-Water Gas Exchange

Sven Eichkorn[1,2], Thomas Münsterer[1,3], Ulrike Lode[1], and Bernd Jähne[1,4]

[1]Institut für Umweltphysik, Universität Heidelberg, Germany
[2]now with Max-Planck-Institut für Kernphysik, Heidelberg, Germany
[3]now with Vitronic Bildverarbeitung GmbH, Wiesbaden, Germany
[4]Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR) Universität Heidelberg, Germany

30.1 Introduction

Anthropogenic climatic changes, caused by an increasing concentration of several trace gases like CO_2 in the atmosphere—the so-called *greenhouse effect*—enhanced public interest in environmental research. A crucial question is whether the ocean can act as a reservoir for such trace gases and dampen the effect of high CO_2 emission. The research field *air-water gas exchange* is dedicated to this topic.

The most important parameter of the air-water gas exchange is the *transfer velocity* k. It indicates the velocity of a hypothetical gas column that penetrates into the water via the air-water interface. It takes several hours to measure k by classical methods. The fluorescence imaging methods described here make it possible to measure k instantaneously.

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

The dependence of k on wind speed and water surface films has been investigated intensively in recent years. But until now the mechanisms controlling this transfer process are not well understood. A thin layer (20-300 μm) at the phase boundary, the aqueous mass boundary layer, limits the exchange because turbulence, which are the most effective transport process can not cross the phase boundary. Several models have been developed to describe the gas transfer theoretically: stagnant film model, small eddy model and surface renewal model. They explain the gas exchange in the boundary layer by molecular diffusion, turbulence or a combination of both. Fluorescence imaging methods of the last years [1, 2] are able to visualize time resolved 2-D concentration profiles of in-water dissolved gases. Thus they give significant insight into the processes of the gas exchange going on in the boundary layer. Direct visualization makes it possible to distinguish between the different models by comparison with the measurements and to alter or develop new models if necessary.

How does the fluorescence technique work? It is based on the visualization of gases that would be invisible when dissolved in water under normal circumstances. A fluorescence indicator dye is diluted in water. A light source excites dye molecules. If the used light source is a laser, the technique is named *laser-induced fluorescence* (LIF, see Volume 1, Chapter 12). Excited molecules relax to the ground state and emit fluorescence light. The intensity of fluorescence emission of the dye is linearly proportional to the local pH value of the solution. Acid or alkaline gases undergo a chemical reaction with water and change the local pH value of the solution. Therefore, the gas is detected by local changes of the fluorescence intensity. It is a direct measure for the concentration of the gas.

In earlier approaches the fluorescence intensity was measured by photocells or photomultiplier tubes. The development of *charge coupled devices* (CCD) opened new opportunities. Now fluorescence intensities are imaged over entire areas. Thus spatial resolution is enormously improved. The transfer velocity k is proportional to the slope of the concentration profile at the air-water interface, which can be extracted from the images.

The first section of this chapter gives a historical review of the fluorescence imaging of air-water gas exchange. The approaches of different research groups are summarized. Image processing tools became very important in the latest experiments of Münsterer [2] and Eichkorn [3]. In the main section of this chapter optical arrangement, image capture, image processing, and the results of those experiments are described. This chapter ends with a summary and a critical discussion of the possibilities and limits of image processing, results, and future goals.

30.2 Historical review

Hiby [4] developed the first fluorescence method to measure the mass transfer at the Rheinisch Westfälische Technische Hochschule Aachen. Many members of the group publicized their results [5, 6, 7]. They examined the absorption of NH_3 in falling-film columns, where a thin liquid film runs down at a vertical wall under the influence of gravity. The fluorescence was excited by a mercury vapor lamp. *Acridin* was used as a *fluorescence indicator* because this dye's emission is independent of the pH value at a wavelength of about 450 nm. At any other wavelength its emission is linearly proportional to the pH value. An optical beam splitter and interference filters allowed detection of the two different wavelengths of the emission spectrum (the pH independent one and a pH dependent one) by secondary electron multipliers. Those two signals and the gas concentration in the air were used to calculate the gas concentration in the fluid phase. Fick's second law allowed calculation of the effective diffusion coefficient. These were used to determine the water-side mass transfer coefficients. The dependence of NH_3 absorption on the Reynold's number of gas and liquid was investigated.

Pankow and Asher [8] and Asher and Pankow [9] described an imaging technique to measure the dependence of the transfer velocity k on the system turbulence. Turbulence of reproducible intensity and scale was generated in a grid stirred tank. With a laser as light source very high sensitivity and spatial resolution were obtained. An argon ion laser beam illuminated the fluorescence of diluted *dichlorofluorescein* (DCFS) in a defined path parallel to the water surface. Silicon photocells recorded the intensity of the laser-induced fluorescence. Assuming a linear concentration profile, the knowledge of the CO_2 concentration at the interface and the CO_2 concentration from the fluorescence intensity signal were sufficient to calculate k.

Asher and Pankow [10] used an argon ion laser beam penetrating the water surface perpendicular to the top. To eliminate the DCFS fluorescence from the bulk a second dye that strongly absorbs the excitation light was added. The fluorescence light at the water surface showed the integrated intensity of light over the depth of the boundary layer. This signal was detected by photocells from the top. The technique allowed for study of the temporal and spatial fluctuations of the CO_2 concentration. The result was that a surface layer prevents the turbulence eddies from completely renewing the gas-liquid interface.

At the University of Illinois, Urbana-Champaign, a different technique was developed. It measures the concentration of dissolved oxygen in water [11]. Oxygen perturbs the excited states of the fluorescent dye, *pyrenebutyric acid* (PBA) and quenches its fluorescent lifetime. The loss of fluorescence light is a function of the dissolved oxygen concentration. A nitrogen laser excited the fluorescence at a wavelength of

337 nm. A CCD camera detected changes of the oxygen concentration simultaneously at different locations along a line perpendicular to the interface. The optical axis was parallel to the interface. The area of interest is magnified by a lens system. For the first time a 1-D concentration profile was visualized. The spatial variation of oxygen was examined. Duke and Hanratty [1] expanded the laser beam by a cylindrical lens to generate a small laser light sheet and to visualize a 2-D concentration profile.

At the circular wind/wave facility, Heidelberg, Münsterer [12] used fluorescein and an argon ion laser to measure 1-D concentration profiles with a high temporal and spatial resolution. Münsterer et al. [13] generated a light sheet by scanning the laser beam with a mirror. The light sheet was used to visualize a 2-D concentration profile parallel to the wind direction and perpendicular to the water surface. Experiments with different gases (HCL, NH_3 and CO_2) were carried out by Münsterer [2] and Eichkorn [3]. For the first time events of surface renewal were detected. An event of surface renewal is an eddy that sweeps through the boundary layer and transports high concentration fluid into the bulk. The transfer velocity and the spatial and temporal variation of the concentration were examined. The experimental setup and the computer vision aspect of those experiments are described in the following section.

30.3 LIF measurements of concentration fields

The circular wind/wave facility at the Institute for Environmental Physics at the University of Heidelberg is a 30-cm wide and 70-cm high gas-tight annular channel with an outer diameter of 4 m. Wind is generated by a rotating paddle ring up to a speed of 10 m/s. Figure 30.1 shows a sketch of a cross section of the channel at the position of the LIF experiments.

The gas exchange between air and water is in equilibrium when the partial pressure of a gas is the same in both media. The same quantity of the gas invades the water and leaves the water per unit of time. To measure the transfer velocity a disequilibrium is necessary. One possibility is to increase the concentration of the gas in the air space. Thus more gas invades the water and the concentration profile of the gas is dependent on the depth. In the experiments described here a defined amount of the gas of interest could be let into the channel via a flowmeter.

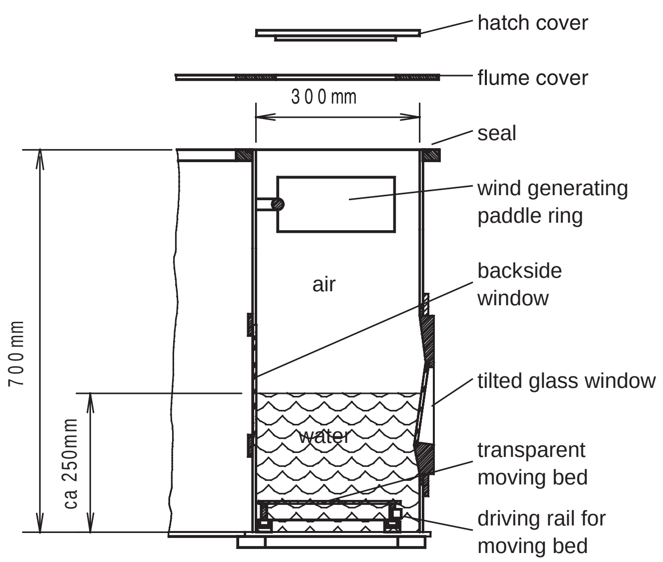


Figure 30.1: *Cross section of the wind/wave flume at the position used in the LIF experiments. The window at the outer side of the channel is tilted to avoid optical distortions.*

30.3.1 Experimental setup

During the LIF experiments the water height was 25 cm. Fluorescein was added to the deionized water to form a 2×10^{-5} M solution.

Figure 30.2 shows the experimental setup for the LIF measurements. Fluorescence is excited by a water-cooled 1 W argon ion laser emitting a wavelength of about 488 nm. The beam travels through a fiber optic cable and a collimator, and is then directed by two mirrors. It pierces the water surface perpendicular from the topside. A Galilean collimator focuses the beam on the water surface, resulting in a spot size of 250 μm diameter. The laser beam is scanned by an optical scanner mirror. The scanned beam forms a 2-D light sheet with a depth of 250 μm and an adjustable width. Scanning the laser beam instead of widening it avoids motion blur. The edges of the light sheet are brighter because they are exposed to the laser beam for a longer time. The best way to get rid of this problem is to generate a light sheet of about 1 cm width and picture only the inner part of the sheet. The scanner mirror has to be in-phase with the camera. The image has to be grabbed when the laser is at its turning point. Thus the laser sweeps over the imaging area just before the image capture and the horizontal brightness is homogeneous.

A digital CCD camera (Dalsa CA-D1) with 256×256 pixel pictures a 4×4 mm region with a frame rate of 200 frames/s. The frames

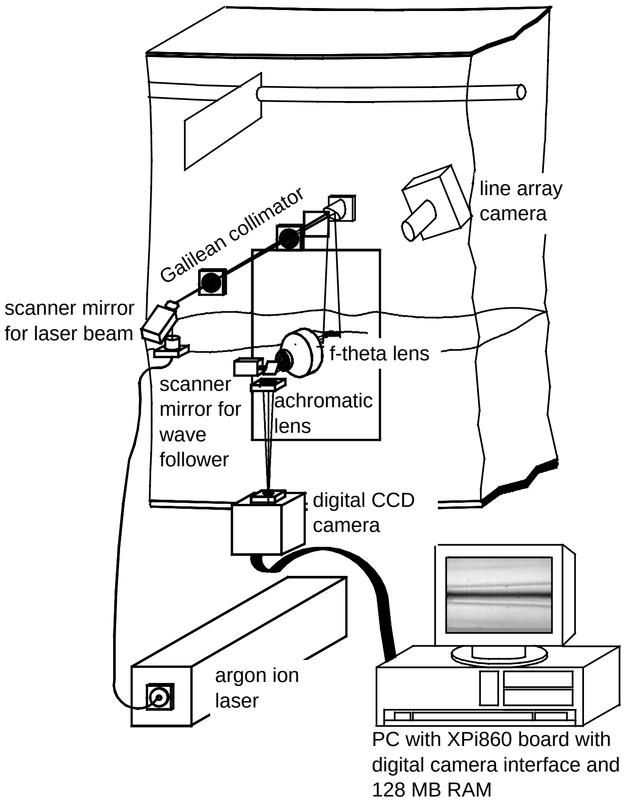

line array camera
Galilean collimator
scanner mirror for laser beam
f-theta lens
scanner mirror for wave follower
achromatic lens
digital CCD camera
argon ion laser
PC with XPi860 board with digital camera interface and 128 MB RAM

Figure 30.2: *Experimental setup: A scanner mirror generates a laser light sheet that pierces the water surface perpendicular from the topside. A 4×4 mm section of the light sheet including the water surface is imaged by a CCD camera through an achromatic lens and an f-theta lens. The complete optical setup is tilted approximately 8° to the horizontal to avoid distortion by adhesion effects at the window. The second scanner mirror and the line array camera are necessary for the optical wave follower (Section 30.3.2).*

were read out digitally via the digital camera interface of a Hyperspeed Technology XPi860 extension board. This board was equipped with 128 MByte of RAM memory allowing acquisition of 1800 frames, a sequence of 9 s. The imaging system consists of an f-theta lens (Section 30.3.2) with a focal length of 25 mm and another achromatic lens of the same focal length resulting in a 1:1 reproduction. Achromatic lenses are optimized for a setup where the image and the object each are in the focal plane of the subsequent lens. Therefore the light sheet and the CCD chip of the camera were in the focal plane of the respective lenses. The optical axis was tilted 8° to the water surface. This was necessary to minimize shading of the measuring section by deeper wave troughs in the foreground. This tilting produces a total reflection that is helpful to find the water surface position by means of image processing. The images were taken through a 8° tilted window to avoid optical blur. The optical resolution was 16 μm, similar to the pixel distance of the camera.

Additional equipment included two PT-100 sensors, which measured the temperature of air and water and a Corning pH meter.

30.3.2 Optical wave follower

In the Heidelberg flume the waves can reach a peak-to-peak height of 10 cm. At a wind speed of only 2 m/s the water surface travels out of the 4×4 mm image section. To measure at higher wind speeds, two different approaches have been followed:

The easier way was to use the surfactant Triton X-100 in a 5×10^{-6} M dilution to suppress waves and turbulence at the water surface. Triton X-100 is a highly soluble surfactant commonly used for gas exchange studies [14]. It generates a monomolecular layer on the water surface, which is no resistance for the gas exchange. Transfer velocities at low wind speeds are found to be the same with and without surfactant. At higher wind speeds the gas exchange rate with surfactant is significantly smaller because turbulences that enhance the gas exchange are suppressed.

Organic monomolecular surface films are often present in coastal waters. Using a surfactant in the lab simulates such circumstances. With Trition X-100, measurements of the gas exchange rate at a smooth interface could be conducted up to a wind speed of 7 m/s.

The second way was to keep the water surface in the imaging range with an optical wave follower. Münsterer [12] first constructed an optically driven wave follower with a scanning mirror in the optical path of the imaging CCD camera. Münsterer [2] used the following setup: to avoid optical distortions caused by the scanning mirror an f-theta objective was used (Fig. 30.2). The f-theta lens was manufactured by Rodenstock, Germany. It is a so-called flat field lens, which describes its task quite accurately. The problem with using a scanning mirror is first that the focal plane is a spherical shell instead of a flat plane (CCD chip). Second, optical distortions beyond a scanning angle of 3° are unacceptably high. The f-theta lens takes care of both of these problems. The focal plane is projected back on a flat field and scanning angles of up to ±25° are acceptable.

A line array camera in the air space on the inside of the flume (Fig. 30.2) pictures a vertical section of the laser light sheet, including the undulated water surface. The output signal of this line array camera is connected to an adjustable comparator thresholding the wave height signal. This comparator gives a 0 V signal for rows read out of the array before a peak was detected (air). When the intensity signal associates with the fluorescence signal (water), the output signal switches to a positive TTL high signal. This TTL signal is connected to a data acquisition board in a second PC. The board was programmed to calculate the wave height from the incoming TTL signal and drive a scanner mirror in the optical path of the imaging optics. Thus the scanning mirror follows the wave motions up to a wave height of 10 cm.

The line array camera triggers all the other equipment after every read-out event. Its frequency was adjusted to the frequency of the image acquisition and to the scanning frequency for the laser light sheet.

30.3.3 Combined fluorescein/oxygen quenching experiments

Münsterer et al. [13] described a very interesting way to visualize profiles of two different gases simultaneously at the same location. Simultaneous measurements help to detect possible systematic errors and offer the chance to learn more about the underlying gas exchange mechanisms.

The fluorescein method described here was combined with the oxygen quenching method mentioned in Section 30.2 [11]. The water contained two dyes, fluorescein and PBA. Experiments showed that the two dyes (the two methods) exhibit a high degree of chemical and optical compatibility. Fluorescein fluorescence is neither affected by the presence of PBA in the solution nor significantly stimulated by the N_2 laser, which excites the PBA fluorescence. The fluorescence of fluorescein also shows only neglectable quenching by oxygen. The PBA fluorescence is not affected by the local pH value of the solution and is not stimulated by the argon ion laser.

The experimental setup shown in Fig. 30.2 was expanded by a nitrogen laser, two dichroic mirrors and an additional camera. The first dichroic mirror allowed the two laser beams to travel the same optical path. The nitrogen laser beam was not scanned, resulting in a 1-D oxygen concentration profile. The pulse rate of the nitrogen laser was 15 Hz. A second dichroic mirror in the imaging optical path in front of the Dalsa CCD camera allowed for capturing the PBA fluorescence (the oxygen concentration profile) with a second camera, a standard NTSC camera.

The time resolution of 15 frames/s of the oxygen quenching technique impedes the investigation of dynamic processes. A thermoelectrically cooled high resolution camera [1] should have been used. The experiments showed that the two techniques cannot be combined when HCL is used as an initiator for the fluorescein method. The PBA fluorescence was seriously quenched in the presence of HCl. Runs with NH_3 as an initiator provided too small transfer coefficients. For a combina-

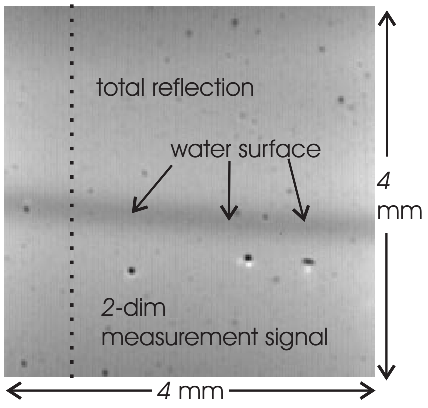


Figure 30.3: *Raw image: a vertical light sheet parallel to the wind direction is pictured. The dark line (highest gas concentration) represents the water surface. The upper part of the image pictures the total reflection and the lower part the 2-D concentration profile. The dark dots are the consequence of dirt on the CCD chip. CO_2 was used as initiator. A 1-D gray-value profile along the dotted line is shown in Fig. 30.4.*

tion of both techniques another acid or alkaline gas has to be found. Measurements with only fluorescein provide good results with HCl and CO_2.

30.3.4 Images and image processing

The result of a measurement run is a sequence of 1800 raw images. In Fig. 30.3 an example of such a raw image is shown. It pictures a 4×4 mm section including the air-water interface, representing a cross section parallel to the wind direction. The irregular dots are the consequence of dirt particles on the CCD chip of the camera. The water surface is represented by a dark line corresponding to the highest CO_2 or HCl concentration. Using NH_3 for alkalinity would illuminate the interface and boundary layer. Without the additional CO_2 concentration in the air space of the channel one would see a nearly homogeneous gray value in the whole picture. The absorption would cause a small decrease in the water depth. The upper part of the image is a total reflection, resulting from those imaging rays that would pierce the laser light plane in the air space. Those rays are reflected at the water surface because of the forementioned 8° angle and thus the image of the laser light sheet is seen. The part below the water surface line is the measurement signal.

The diagram in Fig. 30.4 shows a plot of a 1-D gray-value profile along the dotted line in the raw image (Fig. 30.3). The left part repre-

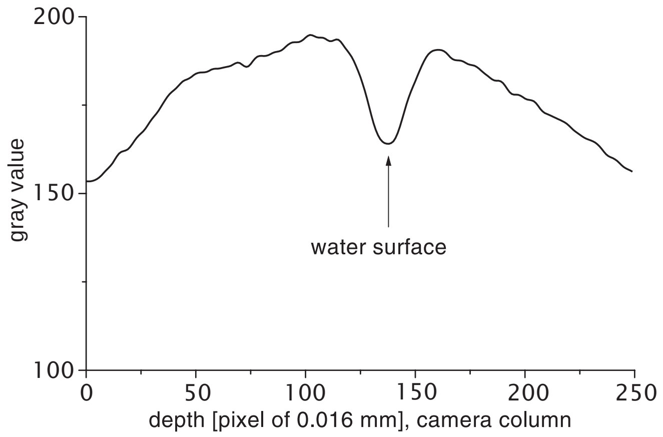


Figure 30.4: *One-dimensional gray-value profile (high gray values indicate high gas concentration) along the dotted line of Fig. 30.3. The left-hand side represents the total reflection. Following the profile on the right-hand side of the water surface the steep increase of the gray value indicates the decreasing gas concentration with water depth in the boundary layer. The smaller decrease of gray value is the result of the absorption of laser light by dye molecules in the water. The gas concentration in this region, called "bulk," is constant.*

sents the total reflection. The position of the water surface is marked. Following the profile to the right the steep increase of the gray value with the water depth in the boundary layer can be seen. This is the interesting signal. The decrease of the gray value deeper in the water signifies the absorption of the laser light by dye molecules. In this bulk area gas concentration is constant. The decrease due to absorption falsifies the measurement signal and has to be corrected.

The image processing was done with the software platform heurisko. It provides standard image processing operators and provides the possibility of developing new operators. Those operators simplify the development of image processing algorithms.

The first aim of image processing was to remove the mentioned dots on the raw images. Dirt particles cause a big contrast in gray value in comparison to the background. The following operator works as a highpass filter: subtracting a smoothed image from the raw image delivers a mask in which the dots are segmented. This mask was eroded to prevent segmentation errors. The next step was to multiply it with a raw image, resulting in an image with gray values of zero at the places where the dots had been and valuable information at all other places. Finally, a normalized convolution was performed with this image, generating a smoothed image without dots. Such a smoothed image is presented in Fig. 30.5. A comparison of the raw image and the smoothed image

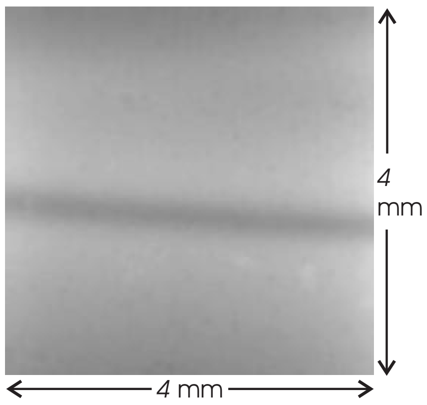


Figure 30.5: *Image of Fig. 30.3 after removing dirt particles with the mentioned smoothing algorithm. The error of this operation is less than 1%.*

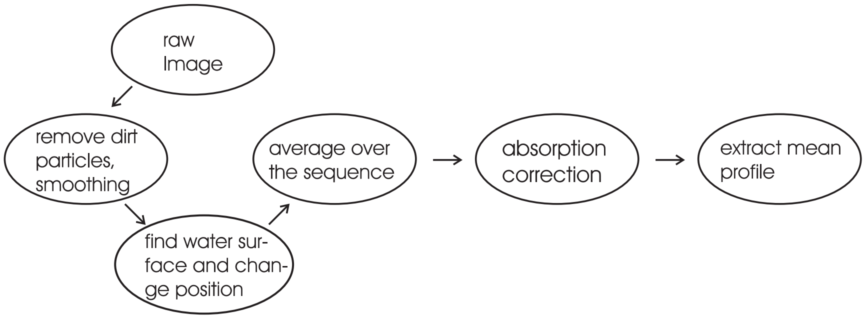

raw Image
remove dirt particles, smoothing
find water surface and change position
average over the sequence
absorption correction
extract mean profile

Figure 30.6: *Image processing steps.*

showed that the measurement signal was falsified by this operation with an error of less than 1 %.

The next section addresses the question of how to find the water surface position. In the 1-D gray-value profile of Fig. 30.4 one can see that the water surface is represented by a local minimum of gray value, due to total reflection. A minimum operator detected the approximate surface position. The exact position of the water surface in each row of the image was found by fitting a parabola in the gray value profile in the area close to the approximated surface position. The surface position was moved to the top of the image. An averaging over the whole sequence was performed with the images processed this way.

Münsterer [2] implemented an algorithm combining symmetry filters and a second derivative finder with fuzzy logic to find the position of the water surface at a wavy interface.

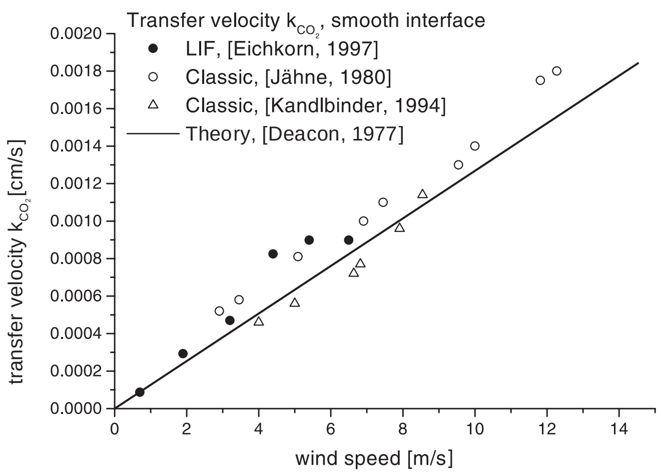


Figure 30.7: *The dependence of the transfer velocity k of CO_2 on the wind speed. Comparison of results of LIF experiments with classical measurements [15, 16] and theoretical studies [17].*

The laser light is absorbed by dye molecules. That is why fluorescence intensity decreases with water depth. This absorption is corrected in the smoothed, averaged images as the last step of the image processing. All steps are summarized in Fig. 30.6. The result of the processing is a mean profile of a sequence. The slope at the water surface was used to calculate the transfer velocity k. Using CO_2 as an initiator makes further calculations necessary because of its slow reaction in the water [3].

The tools presented here to calculate an average transfer velocity of a sequence were complemented by analyses of the temporal development of the concentration profiles in a sequence (see, e.g., Fig. 30.8, detection of surface renewal events) and comparison of the shape of the concentration profiles with profiles predicted by models.

30.3.5 Results

Figure 30.7 shows the gas transfer coefficient k of CO_2 plotted versus the wind speed measured by Eichkorn [3] with LIF compared to a theoretical curve and to classical gas exchange measurements. They correspond well.

The most breathtaking result of the LIF method described here is the direct observation of surface renewal events. No other method before could reveal such an insight into the processes in the boundary layer. Those events are turbulence eddies that nearly reach the water surface

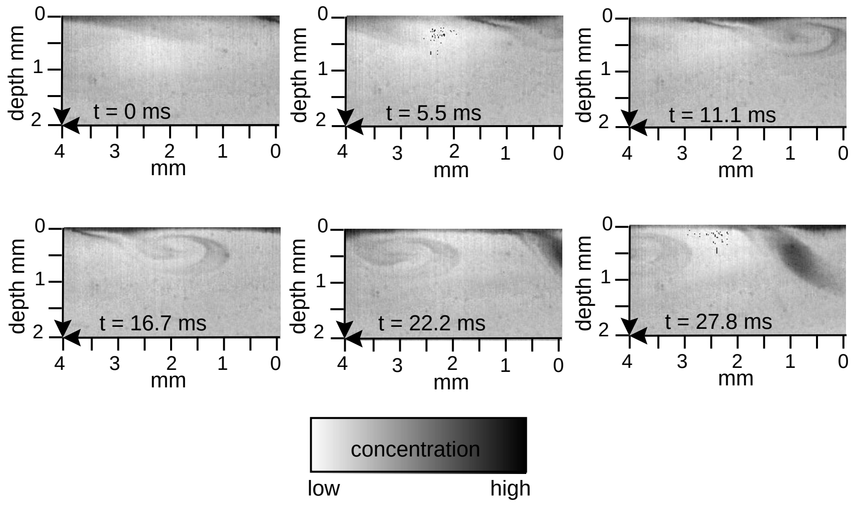


Figure 30.8: *Sequence of LIF images with the wind coming from the right. The time step between the images is 1/180 s. The images are transferred into the frame of reference of the undulated water surface. The sequence shows a large eddy sweeping through the boundary layer transporting high concentration fluid into the bulk. After the first eddy a second one can be seen. Both of these eddies reduce the boundary layer thickness to almost nothing.*

and sweep parts of the water surface and the boundary layer into the bulk and, therefore, replace the gas concentration at the surface by the concentration in the bulk. They occur statistically, and the number of events increases with increasing wind speed. The stagnant film model postulates that no eddies are present in the boundary layer and can therefore clearly be abandoned. The surface renewal model postulates such statistical events and is supported by this observation. In Fig. 30.8 six consecutive images of a sequence are presented. The images are transferred into the frame of reference of the undulated water surface. The sequence shows a surface renewal event. A large eddy sweeps high concentration fluid into the bulk. After the first eddy a second one can be seen.

The shape of the concentration profiles showed a better correspondence with the predictions of the surface renewal model than with those of the small eddy model. The difference in gas exchange mechanisms between a wavy and a flat interface seems to lie in the penetration depth of surface renewal events. The measurements show that for a wavy interface the surface renewal events can penetrate up to the water surface. Although surface renewal is also seen for a flat interface, penetration to the very surface is not observed.

30.4 Critical discussion and outlook

The LIF method utilizes image processing tools for a closer look at nature's processes going on at the air-sea interface. The image sequences give new insight into these processes. The variety of results and conclusions presented could not have been acquired with classical methods. Such successes enhance interest in further investigation. There are several ideas to improve the system.

Future experiments should focus on better resolution and an understanding of the 3-D structure of turbulence effects. A 2-D light-sheet technique cannot completely reveal the spatial structures of turbulent eddies. Small circulation cells with their rotational axis in wind direction were reported by Haußecker [18]. Expanding the current setup into a true volume visualization technique could reveal the nature of these structures and their influence on gas exchange. This is quite complicated but an easy first approach could be to tilt the laser light sheet to an angle of 90°, enlighten a plane perpendicular to the wind speed, and examine the structures in this plane. For both planes experiments with a magnification different than 1:1 could be carried out to visualize a bigger section of the area of interest.

Improving the image processing tools for image sequence analysis could increase the understanding of surface renewal effects. At this point burst phenomena can only be observed qualitatively. With sophisticated image sequence analysis tools tracking and quantification of single effects seem to be possible.

An online calculation of the local transfer velocity could be developed. This would provide the possibility of measurements over a long period of time and experiments with continuous wind variation. The examination of the eddy structure and their dependence on the wind speed needs further investigation.

Jähne et al. [19] propose a variation of the method, which is in development right now. The camera visualizes the water surface from above. The laser light sheet is perpendicular to the water surface so the camera sees a line that consists of the fluorescence light originating from all different depths. A spectrometer analyzes this light. A second dye in the water strongly absorbs in the range of emission of fluorescein. Because of the second dye the shape of the observed fluorescence spectra is dependent on the length of the path the light traveled through the water. With adequate calculations it seems to be possible to obtain depth resolved profiles of the boundary layer. With this new setup simultaneous measurements with the controlled flux technique (CFT, [18]) are feasible.

30.5 References

[1] Duke, S. R. and Hanratty, T. J., (1995). Measurements of the concentration field resulting from oxygen absorption at a wavy air-water interface. In *Air-Water Gas Transfer. Selected Papers From the Third International Symposium on Air-Water Gas Transfer*, B. Jähne and E. Monahan, eds., pp. 627–635. Hanau: Aeon.

[2] Münsterer, T., (1996). *LIF investigation of the mechanisms controlling air-water mass transfer at a free interface.* Dissertation, Institut für Umweltphysik, Universität Heidelberg.

[3] Eichkorn, S., (1997). *Visualisierung und Quantifizierung des CO_2 Gasaustausches mittels laserinduzierter Fluoreszenz (LIF).* Diploma thesis, Institut für Umweltphysik, Universität Heidelberg.

[4] Hiby, J., (1968). Eine Fluoreszenzmethode zur Untersuchung des Transportmechanismus bei der Gasabsorption im Rieselfilm. *Wärme und Stoffübertragung*, **1**:105–116.

[5] Braun, D., (1969). *Die effektive Diffusion im welligen Rieselfilm.* Dissertation, Technische Hochschule, Aachen.

[6] Fahlenkamp, H., (1979). *Zum Mechanismus des Stofftransports im laminarwelligen Rieselfilm.* Dissertation, Technische Hochschule, Aachen.

[7] Hartung, K., (1975). *Untersuchung technischer Mischvorgänge mittels pH-Indikatoren.* Dissertation, Technische Hochschule, Aachen.

[8] Pankow, J. and Asher, W., (1984). Carbon dioxide transfer at the gas/water interface as a function of system turbulence. In *First Intern. Conference on Gas transfer at Water Surfaces*, W. Brutsaert and G. H. Jirka, eds., pp. 101–111. Dordrecht: D. Reidel Publishing Company.

[9] Asher, W. and Pankow, J., (1986). The interaction of mechanically generated turbulence and interfacial films with a liquid phase controlled gas/liquid transport process. *Tellus*, **38B**:305–318.

[10] Asher, W. and Pankow, J., (1988). Direct observation of concentration fluctuations close to a gas-liquid interface. *Chemical Engineering Science*, **44**:1451–1455.

[11] Wolff, L. M., Liu, Z. C., and Hanratty, T. J., (1990). A fluorescence technique to measure concentration gradients near an interface. In *Air-Water Gas Transfer, Selected Papers from the Second International Symposium on Gas Transfer at Water Surfaces*, S. Wilhelms and J. Gulliver, eds., pp. 210–218. Minneapolis, Minnesota: ASCE.

[12] Münsterer, T., (1993). *Messung von Konzentrationsprofilen gelöster Gase in der wasserseitigen Grenzschicht.* Diploma thesis, Institut für Umweltphysik, Universität Heidelberg.

[13] Münsterer, T., Mayer, H., and Jähne, B., (1995). Dual-tracer measurements of concentration profiles in the aqueous mass boundary layer. In *Air-Water Gas Transfer. Selected Papers From the Third International Symposium on Air-Water Gas Transfer*, B. Jähne and E. Monahan, eds., pp. 637–648. Hanau: Aeon.

[14] Frew, N., Bock, E., McGillis, W., Karachintsev, A., Hara, T., Münsterer, T., and Jähne, B., (1995). Variation of air-water gas transfer with wind stress and surface viscoelasticity. In *Air-Water Gas Transfer, Selected Papers from the Third International Symposium on Air-Water Gas Transfer*, B. Jähne and E. Monahan, eds., pp. 637–648. Hanau: Aeon.

[15] Jähne, B., (1980). *Parametrisierung des Gasaustausches mit Hilfe von Laborexperimenten*. Dissertation, Institut für Umweltphysik, Universität Heidelberg.

[16] Kandlbinder, T., (1994). *Gasaustauschmessungen mit Sauerstoff*. Diploma thesis, Institut für Umweltphysik, Universität Heidelberg.

[17] Deacon, E., (1977). Gas transfer to and across an air-water interface. *Tellus*, **29**:363–374.

[18] Haußecker, H., (1996). *Messungen und Simulationen von kleinskaligen Austauschvorgängen an der Ozeanoberfläche mittels Thermographie*. Dissertation, Fakultät für Physik und Astronomie, Universität Heidelberg.

[19] Jähne, B., Cremer, C., Eils, R., Fink, R., Platt, U., Schurr, U., and Stitt, M., (1997). *Fortsetzungsantrag der Forschergruppe zum Studium dynamischer Vorgänge*. Universität Heidelberg.

31 Particle-Tracking Velocimetry

Dirk Engelmann, Michael Stöhr,
Christoph Garbe, and Frank Hering

Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR)
Universität Heidelberg, Germany

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

31.1 Introduction

The need for quantitative flow field analysis has led to the development of various types of instruments. In most cases the measurement techniques were restricted to a single point in physical space, such as *laser doppler anemometry* (LDA. In recent years the rapid advances in computer technology enabled the extension of measuring techniques to two and three dimensions for standard applications.

Flow visualization techniques can be classified from different points of view. First, they can be distinguished by the method to make the flow visible. Either discrete particles or continuous tracers, such as fluorescent dyes using *laser induced fluorescence* (LIF), can be used to mark the flow field. Widely used in biological science (see Volume 1, Chapter 12), fluorescent tracers are also well-suited to observing mixing processes [1, 2, 3] and exchange processes. For an example, refer to *air-water gas exchange* in Chapter 30.

Second, we can differentiate flow visualization techniques by the type of velocity information that is obtained. Some techniques, such as *particle-imaging velocimetry* (PIV, for a review see Grant [4]) and least squares matching techniques [3] yield dense velocity fields. These techniques require, however, either a continuous fluorescent tracer or dense particles. With fluorescent tracers volumetric images must be taken so that only slow flows can be observed.

In contrast, *particle-tracking velocimetry* (PTV) does not yield dense velocity fields. The motion of individual particles is measured directly. In order to be able to track the particles, the particle concentrations must be much less than with PIV techniques. PIV techniques do not evaluate the velocity of individual particles, but correlate small regions between two images taken shortly in sequence.

For comparison of the different techniques it is important to note that Eulerian and Lagrangian flow fields are to be distinguished. *Eulerian flow field* measurements (see Adrian [5]) give the flow field as a function of space and time. This is the flow field that is obtained by PIV technqiues. Particle-tracking velocimetry provides—besides the Eulerian flow field— also the *Lagrangian* representation of the flow field. Each seeding particle is traced along its trajectory within the illluminated area. This path of an individual fluid element as a function of space and time is known as the Lagrangian flow field. The path $\boldsymbol{x}$ of a fluid element can be expressed as a function of the initial starting point $\boldsymbol{x}_0$ and of the elapsed time $t - t_0$

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{x}_0, t - t_0) \tag{31.1}$$

Its velocity $\boldsymbol{v}$ is given by

$$\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x}_0, t - t_0) = \frac{\partial \boldsymbol{x}}{\partial t}(\boldsymbol{x}_0, t - t_0) \tag{31.2}$$

Most PTV techniques use streak photography (see Hesselink [6]) as a tool for the determination of the flow field. The velocity field can be obtained by measuring length, orientation and location of each streak (see Gharib and Willert [7]). The length is commonly calculated from the end points of a streak, where the streaks are detected by a segmentation algorithm. This approach to PTV is only feasible at low particle concentrations of up to typically a few hundred particles/image (see Gharib and Willert [7], Adamczyk and Rimai [8]). Frequently used by many authors is a physical model which employs an interpolation scheme for identifying the same particle in the next image frame; see Section 31.4.

This chapter is restricted to PTV techniques because they yield both Eulerian and Lagrangian flow fields. They are also more suitable than PIV techniques with dense particle concentrations to be extended to true 3-D flow measurements. While standard PIV and PTV techniques give only a snapshot of the 2-D velocity field in a plane, we discuss in Section 31.3 the image processing techniques to track particles over long sequences of images. These techniques do not work with the standard thin laser illumination sheets because the particles leave this thin sheet too quickly to be tracked over an extended period of time. Thus we discuss several alternative illumination techniques in Section 31.2.

PTV techniques result in both Eulerian and Lagrangian flow fields and therefore allow to study the dynamics of the fluid flow. The flow fields still contain only a 2-D velocity restricted to a planar cross section involving one temporal and two spatial coordinates. In contrast, *stereoscopic PTV* (Section 31.4) yields 3-D flow fields in a volume, that is, the complete information of the physical space, which involves three spatial and one temporal coordinates.

31.2 Visualization

The visualization is part of the experimental setup. The aim of the visualization is to make the physical properties of a system optically visible. In the case of flow visualization either seeding particles or bubbles are placed in the fluid and illuminated by a light source with an optical lens system. The CCD camera then records the area of interest. In this way the seeding particles or bubbles represent the flow field of the liquid. An example of such a setup for particle-tracking velocimetry is shown in Fig. 31.4 and will be discussed more detailed in the next section.

The choice of the seeding particles has a major influence on the setup of the experiment. In Section 31.2.4 the properties of transparent seeding particles are discussed.

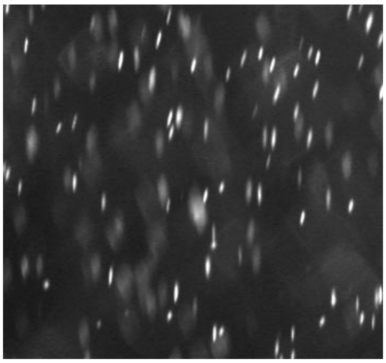


Figure 31.1: *Seeding particles in a light sheet illumination. Due to the exposure time of the camera particles in motion are visualized as streaks.*

31.2.1 Setup for particle-tracking velocimetry

The visualization in classical PTV uses a light sheet which cuts out a small slice of the fluid. The light sheet is typically generated by a laser scanner or by a bright light source with an optical system. For PTV a bright halogen lamp with a cylindrical lens and an aperture is of advantage. Due to the time of exposure of the CCD and the constant illumination, the seeding particles or bubbles are imaged as streak lines (see Fig. 31.1). The streak lines possess properties which are used in image sequence analysis (discussed in Section 31.3.3). Furthermore, the depth of the light sheet has to be chosen in such a way that the seeding particles stay long enough in the illuminated area to enable the tracking. This kind of illumination is only useful if there is a main flow direction and the light sheet is aligned in this direction.

31.2.2 Setup for stereo particle-tracking velocimetry

Flow dynamics in liquids is one of the areas in which spatiotemporal aspects govern the physical properties of the system. Following a tracer—such as seeding particles—over a sequence of images in the three dimensional space of observation yields the complete physical spatiotemporal information. From this the flow field pattern in space and time can then be reconstructed. As compared to the PIV method (see Grant [4]) where vector fields are resolved only for one time interval from a pair of images, the tracking of particles results in the flow field during a larger time span in Lagrangian coordinates.

Stereo PTV extends the classical, 2-D PTV methods to three dimensions. Therefore, the illumination should not be restricted to a light sheet but must fill an entire measuring volume. Typical setups for PTV and stereo PTV applications are presented in the next section.

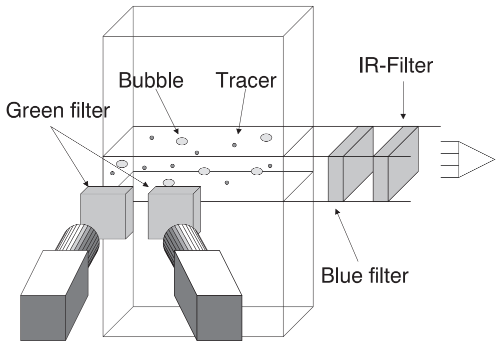


Figure 31.2: Scheme of the water-filled bubble column and visualization for stereo PTV of the flow of water around bubbles with fluorescent tracer particles.

31.2.3 Applications for stereo particle-tracking velocimetry

Generally, PTV methods are applied in applications where dynamic processes are investigated. Sample applications in a gas-liquid reactor and in a wind-wave flume are shown in the next two sections.

Example 31.1: The gas-liquid reactor

In the chemical industry, many process operations depend on complex multi-phase flow processes. A setup for the measurement of the water flow around the bubbles in a gas-liquid reactor is show in Fig. 31.2. Air bubbles move from the bottom to the top. The flow field of the liquid is visualized by fluorescent seeding particles (90 μm diameter) as tracers. The wavelength shift between the blue light exciting the fluorescence and the green fluorescent light makes it possible to suppress the light directly reflected from the bubbles with an appropriate colored glass filter. The motion of the bubbles can be measured with a second setup (not shown in Fig. 31.2) using a blue filter that passes only the blue light of the light source.

Example 31.2: 2-D flow visualization in a wind-wave flume

Another sample application of PTV is the study of the water flow close to the air/water interface in a wind-wave flume. The Heidelberg wind-wave flume has an annular shape. As shown in Fig. 31.3, the wind is produced by paddles driven by a belt. Due to the wind-induced momentum and the annular shape of the flume, a great average speed in the direction of wind is induced upon the water. This average speed can be compensated for by a moving bed, installed at the bottom of the flume and rotating in the opposite direction of the main flow.

A typical experimental setup for 2-D flow visualization in this facility is shown in Fig. 31.4. Due to their movement during the exposure time, the particles are imaged as streaks (Fig. 31.1). In order to track the particles over an extended period of time, the standard thin laser illumination sheets for particle-imaging velocimetry are not suitable even

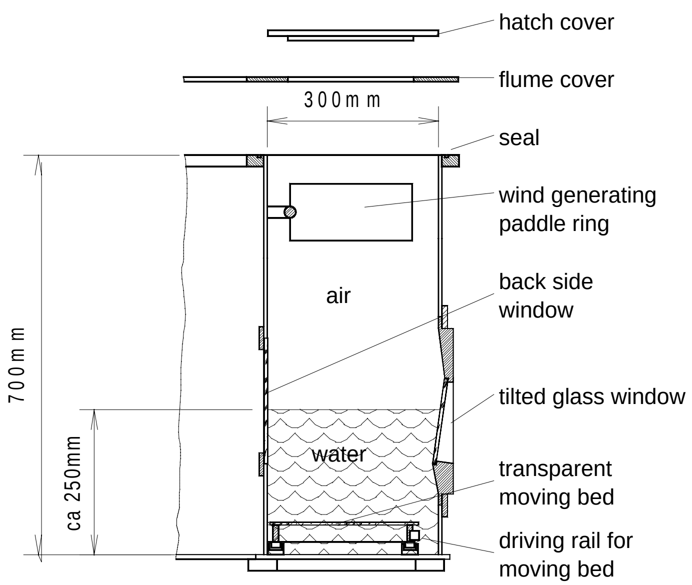


Figure 31.3: *Scheme of the cross section of the circular Heidelberg wind-wave flume.*

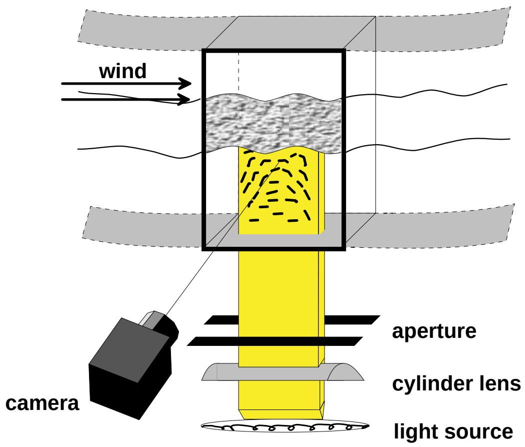


Figure 31.4: *Scheme of the experimental setup for 2-D PTV.*

if the light sheet is aligned with the main flow direction. Therefore, a halogen light source with a cylinder lens and an adjustable aperture to control the width of the illumination sheet is used. This illumination setup is also suitable for 3-D particle tracking with stereo imaging (see Netzsch and Jähne [9]) using a setup similar to that shown in Fig. 31.2.

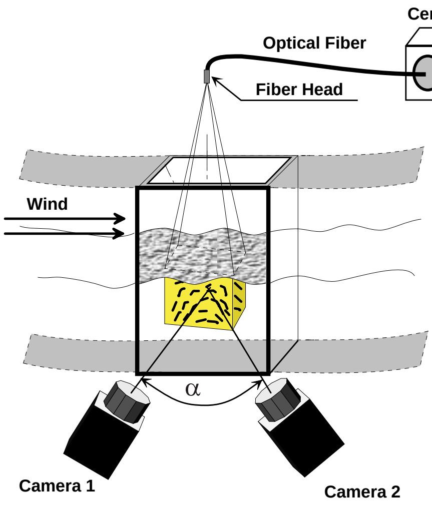


Figure 31.5: *Scheme of the experimental setup for volumetric 3-D PTV with forward scattering.*

Example 31.3: Volumetric flow visualization in a wind-wave flume

An alternative illumination setup for 3-D PTV is shown in Fig. 31.5. Here, a light source with an optical fiber is mounted on the top of the flume, opposite to the CCD cameras. This setup has two advantages over the sheet illumination techniques as discussed in Example 31.2. First, a whole volume can be illuminated. Second, it has a much lower scattering angle than the setups in Figs. 31.2 and 31.4. While the scattering angle is about 90 degrees in these setups, forward scattered light with a scattering angle between 20 degrees and 40 degrees is observed with the setup in Fig. 31.5, resulting in much brighter particle images (see Section 31.2.4).

A problem with this setup is the oblique view of the cameras through the bottom of the facility. This results in a significant lateral *chromatic aberration* (Volume 1, Section 4.5.6). Thus the bandwidth of the polychromatric xenon illumination source has to be reduced by using a red color filter. The observation volume results from the geometry of the two intersecting volumes of the two cameras. The intersecting volume (as a function of angle α) is therefore smaller than that covered by a single camera. In the setup shown in Fig. 31.5 it is about $5\,\text{cm} \times 5\,\text{cm} \times 3\,\text{cm}$.

31.2.4 Choice of seeding particles and scattering angle

Measurements of flow fields in a transparent medium such as water or air require tracers in order to represent the fluid motion correctly [10]. The ideal particle should have two properties. It should follow the flow of the fluid without any slip and delay and it should scatter as much light as possible into the observation direction. Thus it is necessary to investigate both the flow around the particles and the scattering characteristics.

The flow-following properties of a particle depend on its size and density. Assuming spherical particles and neglecting the interaction between individual particles, the vertical flow induced by buoyancy flow can be derived and results in

$$\boldsymbol{v}_d = \frac{(\varrho_p - \varrho_f) d_p^2}{18\mu} \boldsymbol{g} \tag{31.3}$$

where d_p is the particles' diameter, μ is the dynamic viscosity, and $\boldsymbol{g}$ is the gravitational constant. The vertical drift velocity depends on the difference of the densities of fluid (ϱ_f) and tracer (ϱ_p). In order to minimize this vertical drift, the particles density should be equal to the density of the medium. While this can be achieved quite easily for flow in liquids, it is hardly possible for flow in gases.

A general equation of motion for small, spherical particles was first introduced by Basset, Boussinesq and Ossen; see Hinze [11] for details. Analysis of this equation shows how particles behave in a fluctuating flow field at frequency ω; the error $\epsilon_f(\omega)$ is then given by

$$\epsilon_f(\omega) = \text{const.}\, \omega^2 d_p^4 \tag{31.4}$$

For the evaluation of the proportionality constant see Hinze [11]. This error can be reduced by choosing small particles for the flow visualization, but because the visibility of particles is proportional to d_p^2, larger particles are better suited for flow visualization.

Wierzimok and Jähne [12] examined various particles for application in tracking turbulent flow beneath wind-induced water waves. The authors found LATEX-polystyrol particles to be suitable for this purpose. These particles are cheap and have a density of 1.047 g/cm^3. For particle diameters between 50 to 150μm, the drift speed velocity ranges from 2.5×10^{-3} to 0.6 mm/s and the error $\epsilon_f(\omega)$ is below 0.07 %.

Polychromatic light scattering by tracer particles. There are three major aspects which have to be considered:

1. The light source;
2. the angle of observation relative to the light source (scattering angle);
3. and the type of tracer particles.

For PTV the seeding particles are illuminated either by a coherent monochromatic laser sheet or an incoherent polychromatic light source. The wavelength range of such a light source is often reduced by an optical bandpass filter. It is obvious that the scattering properties of particles is influenced both by the coherency and the bandwidth of the light source. These effects are taken into account by the scattering theory. Comparison between the experimental data and the theoretical intensity distributions agree well with Mie's scattering theory. Here we only report the main results. For more details see Hering et al. [13, 14].

Different types of frequently used seeding particles were investigated such as polystyrol, hollow glass spheres, and polycrystalline material. For all investigated types it was discovered that the use of polychromatic light has a strong influence on the curve form. Whereas the intensity distribution of monochromatic light varies strongly with the wavelength, the curve for polychromatic light shows a smooth behavior around a scattering angle of 90°. The averaged distribution varies slightly whereas the fractions of high frequencies are reduced almost completely. This is in accordance with the theoretical calculated values (see Hering et al. [13, 14]).

The investigated seeding particles show an almost constant and smooth scattering intensity of polychromatic light in a certain range of the scattering angle around 90°. For polystyrol seeding particles (30 to 200 μm in diameter) the range is from 55 to 140°, for hollow glass spheres (about 35 μm in diameter) from 80 to 125° and for polycrystalline material (about 30 μm in diameter) from 80 to 150°.

The hollow glass spheres show a greater increase in scattering intensity for angles greater than 125° (up to a factor of three) than the polystyrol and polycrystal particles. A great variation of scattering intensity for all types shows for angles smaller than the region of almost constant scattering intensity.

If small seeding particles are required, a maximum of scattering intensity is desired. The largest scattering intensity is acquired in forward scattering, with an angle less than 55° in the case of the polystyrol seeding particles and also for hollow glass spheres and polychrystalline particles less than 80 μm. Although in this range the variation of intensity is large, this is the recommended scattering angle to gain a maximum of light intensity in the CCD camera sensors. Continuity of optical flow of the gray values does not exist due to the great variation of the scattering angle and therefore can not be taken into account by the PTV algorithm and the correspondence search (see Section 31.3.3).

31.3 Image processing for particle-tracking velocimetry

Several steps of image processing are required for the extraction of the flow field from the image sequences. First the particles must be segmented from the background. This is one of the most critical steps. The better the *segmentation* works, the more dense particle images

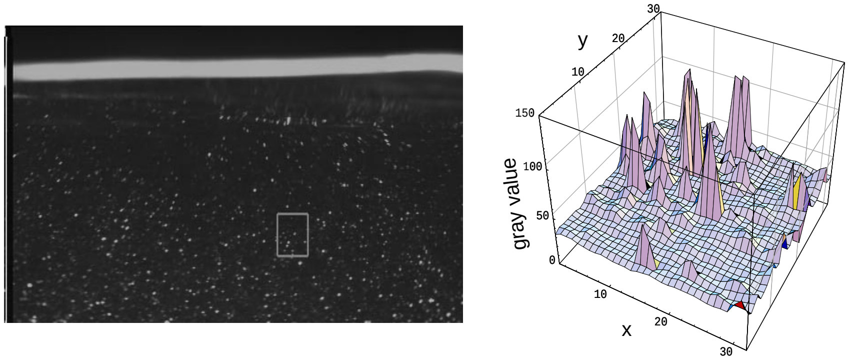


Figure 31.6: *Pseudo 3-D plot of marked area. Streaks can clearly be identified as local maxima in the gray-value distribution.*

can be evaluated resulting in denser flow fields. In this section two different segmentation algorithms are discussed: region-growing (Section 31.3.1) and model-based segmentation (Section 31.3.2). The basic algorithms for tracking segmented particles are discussed in Section 31.3.3, while the removal of bad vectors from the flow field and the accuracy of the algorithms are the topics of Sections 31.3.4 and 31.3.5, respectively. Finally, Section 31.3.6 deals with algorithms that combine the advantages of PTV and PIV.

31.3.1 Segmentation: region-growing

The intensities (gray values) of the streak images show a great variation from very low to very high values. Simple pixel-based segmentation techniques cannot be chosen since the streak images do not exhibit a true bimodal distribution in the gray-value histogram. A ‘region-growing’ algorithm was developed in order to discriminate individual particles from the background. Regions with similar features are identified and merged to a “connected object.”

The image $g(x,y)$ is scanned for local maxima in the intensity, since the location of streaks is well approximated by a local maximum $g_{\max}(x,y)$ (Fig. 31.6). A minimum search horizontally and vertically from $g_{\max}(x,y)$ enables the calculation of the peak height

$$\Delta g = \min(g_{\max} - g_{\min}) \tag{31.5}$$

where $g_{\min}$ are the minima as revealed by a minimum search. In addition, the half-width is measured. Both peak height and half-width are

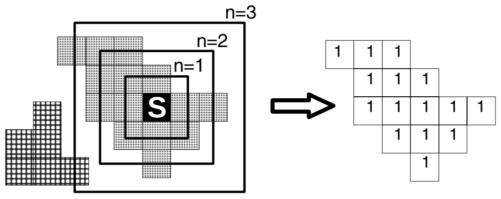


Figure 31.7: *The "region growing algorithm" will grow around the seeding point **s** until no further object points are detected.*

required to keep seeding points above a threshold. In this way random noise is not mistaken as seeding points for the region-growing algorithm.

With the seeding points identified, the region-growing algorithm segments the object (Fig. 31.7) according to the following two rules:

- a pixel to be accepted as an object point requires a gray value higher than an adaptive threshold. This is calculated by interpolation from g_{min}. For details regarding computation of the threshold, see Hering [15]; and
- only pixels which form a connected object are considered.

An example result of the described segmentation algorithm is shown in Fig. 31.8. Each object identified by the segmentation is labeled with a flood-fill algorithm borrowed from computer graphics. The size of each object can then be determined, and as a result large objects (reflections at the water surface) can be removed.

31.3.2 Segmentation: model-based

The previously described region-growing algorithm yields good results as long as the particles have an approximately spherical shape. In this case a plain local maximum describes their center. However, with increasing velocity the particles are imaged as *streaks* because of their motion during exposure time. The central peak is blurred and can no longer be reliably used as a seeding point for region growing since the algorithm tends to break one streak into several smaller ones. An appropriate algorithm therefore has to take into consideration the knowledge of the streak structure.

Modeling the gray-value distribution of a streak. As shown in Hering et al. [14] the light distribution (the gray-value distribution $g_\sigma(\boldsymbol{x})$,

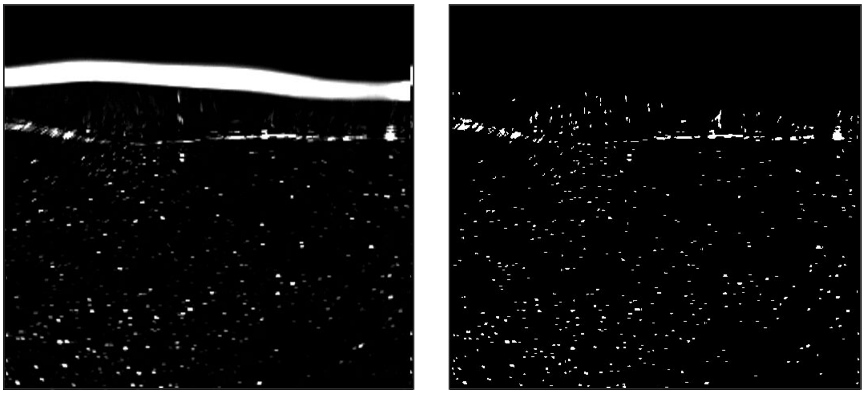


Figure 31.8: Original gray-value image left and segmented image right. The white line on the top of the left image is due to reflections at the water surface and is eliminated by the labeling algorithm on the right image.

respectively) of a still particle can be approximated by a Gauss function which depends only on the distance to its center of mass $\boldsymbol{\mu}$

$$g_\sigma(\boldsymbol{x}) = g_\sigma(|\boldsymbol{x} - \boldsymbol{\mu}|) \text{ with } \boldsymbol{\mu} = \frac{\int \boldsymbol{x} g_\sigma(\boldsymbol{x}) d^2x}{\int g_\sigma(\boldsymbol{x}) d^2x} \tag{31.6}$$

with σ denoting the distribution half-width. To take into account the motion during the exposure time T (assuming a constant velocity $\boldsymbol{v}$), Eq. (31.6) has to be integrated over time and thus the equivalent term for the streak distribution $g_s(\boldsymbol{x})$ is obtained

$$g_s(\boldsymbol{x}) = \frac{1}{T}\int_0^T \mathcal{G}_\sigma^{(2)}(\boldsymbol{x} - \boldsymbol{v}t)dt \tag{31.7}$$

with $\mathcal{G}_\sigma^{(n)}(\boldsymbol{x}) = 1/(\sqrt{2\pi}\sigma)^n \exp\left(-\boldsymbol{x}^2/2\sigma^2\right)$ representing the n-dimensional Gauss function. With the abbreviation $\mathcal{G}(x) = \mathcal{G}_\sigma^{(1)}(x)$ for $\sigma = 1$, Eq. (31.7) is found to be equivalent to (for an explicit calculation, see Leue et al. [16])

$$g_s(\boldsymbol{x}') = A\frac{\mathcal{G}(\frac{1}{\sigma}|\boldsymbol{x}' \times \boldsymbol{n}|)}{l\sigma} \int\limits_{\frac{1}{\sigma}(\boldsymbol{x}'\boldsymbol{n} - \frac{l}{2})}^{\frac{1}{\sigma}(\boldsymbol{x}'\boldsymbol{n} + \frac{l}{2})} \mathcal{G}(\tau)d\tau \tag{31.8}$$

where $\boldsymbol{x}'$ denotes the spatial coordinate relative to the center of mass $\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{\mu}$; $\boldsymbol{n}$ is a normalized vector pointing in the particle direction of motion $\boldsymbol{n} = \boldsymbol{v}/|\boldsymbol{v}|$,and l is the length of the streak that is the distance the particle moved during exposure time. The normalization has been chosen in such a way that the parameter A describes the total sum of the gray value. The width of the streak—normal to its direction of motion—is identical to the width σ of the still particle.

The goal is to determine these parameters with sub-pixel accuracy. This can be done by applying a least square fit of the function Eq. (31.8) to the image data. First, an area of interest is selected by the subsequent preprocessing algorithm. For this area the parameters are estimated and improved iteratively.

Preprocessing. The streaks in the image sequences are characterized by a nearly constant gray value in the direction of motion, whereas they vary rapidly in the perpendicular direction. In order to find an area that contains a streak an algorithm that emphasizes such structures is needed.

Image processing provides us with an appropriate procedure called *local orientation* (for detailed reading on this subject, see Jähne [17] and Volume 2, Chapter 10). By simply applying linear filter operations and point-wise multiplication the following three characteristics of an image can be computed for each pixel:

- the **orientation angle** lies in the range of 0 to 180° and specifies the angle that an ideally oriented structure of parallel lines would enclose with the x-axis. The fact that the angle is limited to an interval of 180° shows that it is only possible to determine the orientation of a structure but not the direction;
- the **amplitude** gives a measure for the certainty of the orientation angle; and
- with the **coherence measure** the *strength* of the orientation can be described. For an isotropic gray-value distribution this measure is zero; for an ideally oriented structure it is one.

To find regions in the image data which contain 'streaks' the coherence measure is the most effective one. It intensifies oriented structures that will be identified as streaks. Moreover, it turns out to be almost independent of the gray-value amplitudes of the streaks.

Segmentation. After calculating the coherence measure the image is binarized using a threshold. Because of the independence on the gray-value amplitude of this measure more than 90 % of the streaks can be detected even though their intensities differ widely (see Fig. 31.9) within the image.

To determine the exact streak parameters (center of mass, length, half-width, orientation, and gray-value sum) Eq. (31.8) is fitted to each binarized region by using the Levenberg-Marquardt algorithm for non-linear regression problems [18]. Start parameters are found by calculating the region moments (see Jähne [17]). Figure 31.10 shows an example result of the model-based segmentation algorithm.

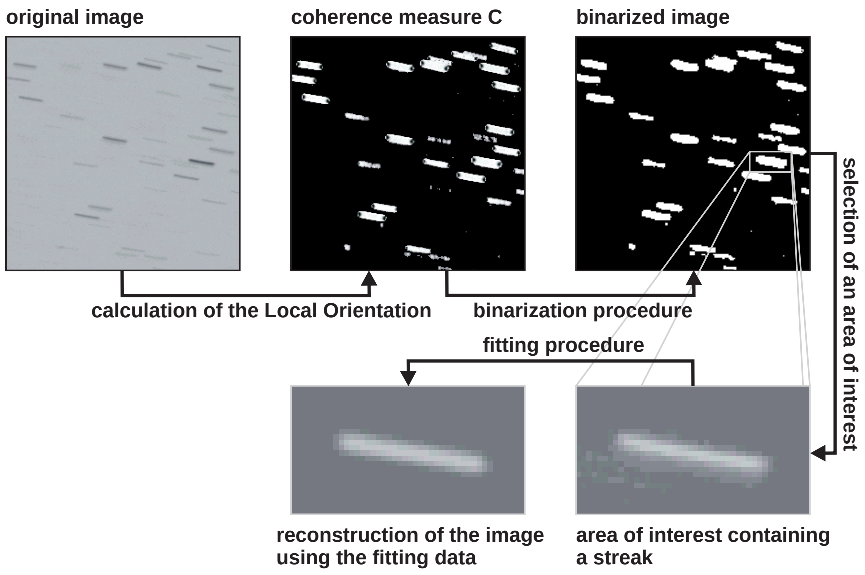

original image
coherence measure C
binarized image
calculation of the Local Orientation
binarization procedure
fitting procedure
selection of an area of interest
reconstruction of the image using the fitting data
area of interest containing a streak

Figure 31.9: *Segmentation procedure as described in the text. On the original image the coherence measure of the local orientation is calculated and then used as an input for the binarization procedure. In a subsequent step the parameters are determined by fitting Eq. (31.8) to the image data.*

31.3.3 Image-sequence analysis

After segmentation, the *correspondence problem* of identifying the same particle in the next image frame is solved. Standard video cameras operate in a so-called *interlaced-scanning* mode (see Volume 1, Section 7.7.1). One frame is composed of two fields—one with the even, the other with the odd lines that are exposed in sequence. Thus with moving objects two different images are actually obtained with half the vertical resolution. In interlace mode, there can be an overlap in the exposure time of the two fields when each field is exposed for the full frame time (so-called frame integration mode). The overlap in the exposure time yields a spatial overlap of the two corresponding streaks from one image to the next (as illustrated in Fig. 31.11) that makes it easy to solve the correspondence problem.

However, the interlaced mode with only half the vertical resolution presents a disadvantage for particle tracking. Thus it is more useful to use modern *progressive-scanning* cameras with full vertical resolution (Volume 1, Section 7.7.3). Then corresponding particles will no longer overlap. This expansion can be increased by the use of a morphological *dilation* operator (see Wierzimok and Hering [19] and Volume 2, Sec-

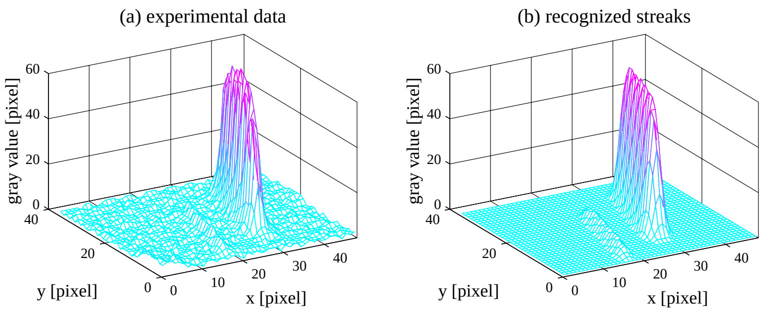


Figure 31.10: *Comparison between the experimental data and the gray-value distribution according to Eq. (31.8):* ***a*** *pseudo 3-D plot of the original data containing streaks of different intensities;* ***b*** *reconstruction of the streaks by the described algorithm. The detection of the two streaks was only possible because the coherence measure is widely independent of the gray-value amplitude.*

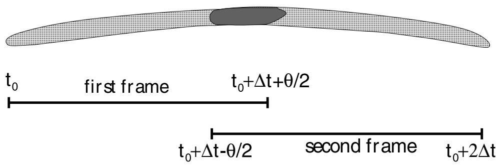


Figure 31.11: *The temporal overlap of the exposure time in two consecutive fields of the same frame yields a spatial overlap of corresponding streaks. A similar effect for images which do not overlap in time is obtained for streaks increased by the use of a morphological dilation operator.*

tion 21.3.1). This operation will enlarge objects and typically smooth their borders (see Jähne [17].) The correspondence is now found by an AND operation between two consecutive segmented fields. This correspondence of streak overlap can be calculated quickly and efficiently (see Hering [15]). In addition, because the temporal order of the image fields is known, the direction of the vector is also known and no directional ambiguity has to be taken into account.

To avoid unnecessary clustering of the objects the dilation is not calculated simultaneously for all objects in an image but for each object individually (see Hering et al. [20]). In most cases, in particular for low particle concentration ($\leq$ 300 particles/image), each particle shows only the overlap with a corresponding particle in the next frame. However, at higher particle concentration, particles show overlap with typically up to four particles in the next frame. Therefore, additional features are required to minimize false correspondences. Ideally, the sum of gray values for each streak S in the image series should roughly be constant due to the equation of continuity for gray values (see Hering [15]):

$$\sum_{x,y \in S} g(x,y) = \text{const} \tag{31.9}$$

This implies that a particle at low speed is visualized as a small bright spot. The same particle at higher speed is imaged as a fainter object extending over a larger area. The sum of gray values in both cases should be identical. The computation of the sum of gray values, however, is error-prone due to segmentation errors which are always present. Therefore, it is more convenient to normalize the sum of gray values by the occupied area. The normalized sum of gray values being G_n^1 of the first frame and G_n^2 of the second are required to lie above a threshold of the confidence interval C

$$C \quad = \quad 1 - \frac{|G_n^1 - G_n^2|}{|G_n^1 + G_n^2|} \quad \longmapsto \quad [0,1] \tag{31.10}$$

A similar expression can be derived for the area of the objects.

Calculation of the displacement vector field. Wierzimok and Hering [19] showed that the center of gray value $\boldsymbol{x}_c$ of an isotropic object represents the time-averaged 2-D location $\langle \boldsymbol{x} \rangle_{\Delta t}$. Thus

$$\boldsymbol{x}_c = \langle \boldsymbol{x} \rangle_{\Delta t} \tag{31.11}$$

where $\boldsymbol{x}_c$ is calculated from the sum of all n segmented pixels of a streak

$$\boldsymbol{x}_c = \left(\frac{\sum\limits_{i=1}^{n} x_i g(x_i, y_i)}{\sum\limits_{i=1}^{n} g(x_i, y_i)} \quad , \quad \frac{\sum\limits_{i=1}^{n} y_i g(x_i, y_i)}{\sum\limits_{i=1}^{n} g(x_i, y_i)} \right) \tag{31.12}$$

The knowledge of the location of the same particle in the previous frame (at the time $t - \Delta t$) allows it to apply the first-order approximation of the velocity field $\boldsymbol{u}(t)$

$$\boldsymbol{u}(t) \approx \frac{\boldsymbol{x}_c(t) - \boldsymbol{x}_c(t-1)}{\Delta t} \tag{31.13}$$

Repeating the described algorithm will automatically track all encountered seeding particles from one frame to the next.

Concepts for minimizing false correspondences. The expected position of a particle is predicted by extrapolation from the vector field of previous time steps (see Wierzimok and Hering [19]). A χ^2−test evaluates the probability that a pair of particles matches. Minimizing χ^2 will maximize the likelihood function. This technique is especially helpful when more than one overlap is encountered. By this method the most unlikely correspondences can be eliminated immediately. Using $\boldsymbol{x}_c(t)$ as a measure for the particle's average position in an image, the motion of the particle center from image to image is a discrete time series described by a Lagrangian motion; f represents the particle trajectory, with its initial location at $t = 0$. Therefore

$$\boldsymbol{x}_c(t) = f(\boldsymbol{x}_c(0), t) \tag{31.14}$$

The function f is usually not known *a priori* and in most flow conditions can only be approximated by a piecewise-algebraic function for a short period of motion. Depending on the degree of freedom n of $f^{(n)}$, it requires n previous images in order to estimate the particle location in the subsequent one. The expected position of the streak centers $\boldsymbol{x}_e(t)$ is thus estimated by

$$\boldsymbol{x}_e(t) \approx f^{(n)}(\boldsymbol{x}_c(t), ..., \boldsymbol{x}_c(t-n+1)) \tag{31.15}$$

Two simple functions f have been implemented to predict the particle position (see also Wierzimok and Hering [19] for details). The first one ($n = 1$) assumes a constant particle velocity; the second one ($n = 2$) takes into account a constant acceleration. This can be written as a convolution (denoted by the $*$ in the following equations) with the kernel h working separately for each component of $\boldsymbol{x}_c(t)$

$$f^{(1)} = h^{(1)} * \boldsymbol{x}_c \quad \text{with} \quad h^{(1)} = [2, -1] \tag{31.16}$$

$$f^{(2)} = h^{(2)} * \boldsymbol{x}_c \quad \text{with} \quad h^{(2)} = [5, -4, 1] \tag{31.17}$$

Now the expected position $\boldsymbol{x}_e(t)$ can be compared to the calculated position $\boldsymbol{x}_c(t)$ and hence multiple correspondences can be reduced to only one.

For further optimization a fuzzy logic approach similar to that of Etoh and Takehara [21] has been chosen to maximize the possibility of finding the same particle in the next frame.

31.3.4 Removing bad vectors from the flow field

In addition to avoiding *multiple correspondences*, false or stray vectors are eliminated by a grid-interpolation technique. Note that the technique described in the following is often used for interpolation of PTV data onto a regular grid (see Agüí and Jiménez [22]). Although this technique seems to be rather crude, only minor enhancements are being made by more elaborated techniques, such as a 2-D thin spline interpolation (STS) described by Spedding and Rignot [23]. Adaptive

Gaussian windowing (AGW) is simple convolution of the vector field with a Gaussian kernel, yielding the interpolated vector field $\boldsymbol{u}_i(\boldsymbol{x}_i)$

$$\boldsymbol{u}_i(\boldsymbol{x}_i) = \frac{\sum\limits_{j=1}^{n} \boldsymbol{u}_j(\boldsymbol{x}_j)\,\exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)}{\sigma^2})}{\sum\limits_{j=1}^{n} \exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{\sigma^2})} \tag{31.18}$$

where σ is the $1/e$–width of the convolution window. Agüí and Jiménez [22] found through Monte Carlo simulations that the optimal width of the window is directly proportional to the mean nearest neighbor distance δ

$$\sigma = 1.24\,\delta, \quad \text{with} \quad \delta = \sqrt{\frac{A}{\pi n}} \tag{31.19}$$

where A denotes the image area, with n segmented particles. This interpolation can be improved and speeded up considerably by localizing the AGW convolution. That means it is not summed up over all vectors as in Eq. (31.18), but only those vectors are taken into account which lie in a certain neighborhood. By choosing the $\boldsymbol{x}_i$ of Eq. (31.18) on a regular grid, the PTV vectors are interpolated to that regular grid. In addition, this interpolation technique can be used to remove stray vectors from the vector field. Each displacement vector gained by the PTV algorithm is compared to the interpolated vector at the same location, obtained by the AGW technique. Obvious false vectors are being eliminated as a consequence.

31.3.5 Accuracy of particle-tracking velocimetry

For testing the *accuracy* of the PTV algorithm, small particles were attached to a rotating disk of an LP-player (see Hering et al. [20]). The disk rotated at constant speed of 33 rpm; see Fig. 31.12. Therefore, as each vector of a trajectory has the same absolute velocity, one can calculate the one-sigma error $\sigma_{\overline{v}}$ for the determination of a displacement vector by

$$\sigma_{\overline{v}} = \sqrt{\frac{\sum\limits_{i=1}^{N} (\overline{v_i} - ||\boldsymbol{v}_i||)^2}{N(N-1)}} \tag{31.20}$$

where $\overline{v_i}$ is the mean displacement, $||\boldsymbol{v}_i||$ is the ith displacement in the trajectory, and N is the number of displacement vectors in the trajectory. Figure 31.13 shows the standard error for displacement vectors of up to 12 pixels per frame, calculated from more than 100,000 vectors. The relative error remains always well below 3 %.

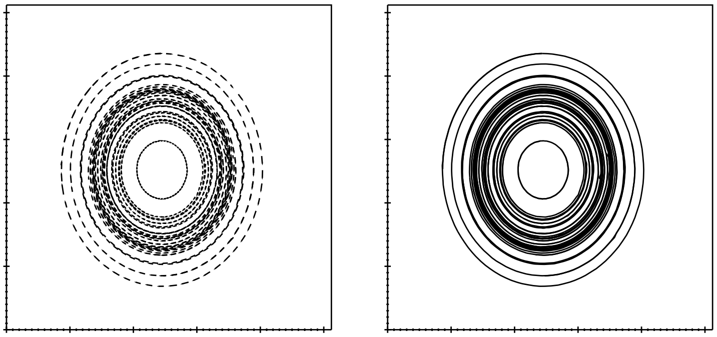


Figure 31.12: *Trajectories of particles attached to a disk rotating at 33 rpm; in the left image only every second vector is shown.*

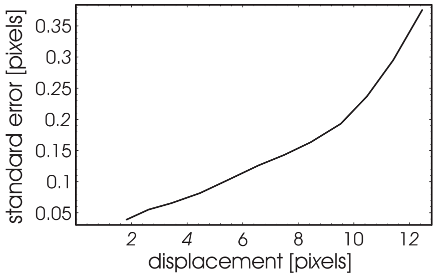


Figure 31.13: *One-sigma error for the calculation of displacement vectors.*

Multiscale particle-imaging velocimetry. The most widely used technique in modern computer-based flow visualization is based on a simple *cross correlation* of two images taken shortly after each other (PIV). A typical application records particles in a single plane within a densely seeded flow field using a pulsed laser sheet; see Fig. 31.1. The illuminated particles are imaged by a CCD sensor, taking at least two consecutive images. A sliding interrogation window of the first image $g(\boldsymbol{x})$ is matched with the second image $h(\boldsymbol{x})$ using the (unbiased) cross correlation $\Phi(\boldsymbol{s})$

$$\Phi(\boldsymbol{s}) = \frac{\sum\limits_{\boldsymbol{x}\in\epsilon} (f(\boldsymbol{x}+\boldsymbol{s}) - <f(\boldsymbol{x})>)(g(\boldsymbol{x}) - <g(\boldsymbol{x})>)}{\sqrt{\sum\limits_{\boldsymbol{x}\in\epsilon} (f(\boldsymbol{x}) - <f(\boldsymbol{x})>)^2}\sqrt{\sum\limits_{\boldsymbol{x}\in\epsilon} (g(\boldsymbol{x}) - <g(\boldsymbol{x})>)^2}} \tag{31.21}$$

This equation is effectively implemented via the 2-D fast Fourier transform (FFT) of the two image samples and a complex conjugate multiplication. Finally, the inverse transformation yields the standard

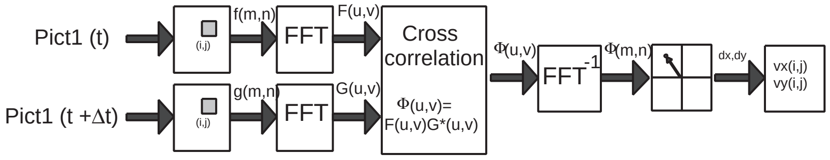


Figure 31.14: *Scheme for the calculation of the cross correlation, as used for digital imaging velocimetry.*

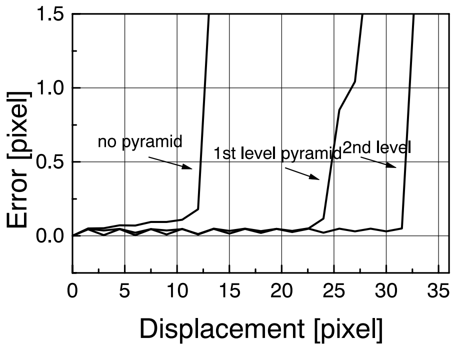


Figure 31.15: *Error of multi-scale PIV with a 32× 32 wide correlation window as a dependency of the pyramid level.*

cross correlation. With a peak-fitting routine the most likely displacement (highest correlation) is detected (see Fig. 31.14).

Various techniques for peak finding are reported in the literature (for a review, see Gharib and Willert [7]). Typical displacements correctly extracted are in the order of half the size of the correlation window ϵ (see Fig. 31.15). In order to extend this technique to larger displacements a new *multiscale* approach will be presented in this chapter that is similar to the techniques described in Volume 2, Chapter 14.

The original images are decomposed in their *Gaussian pyramids* (see Burt and Adelson [24] or Volume 2, Section 4.4.2). On each level (going from the coarse structures to the finer structures) the cross correlation is computed. These displacements are serving as a velocity estimator for the next (finer) pyramid level. This technique allows a drastic increase in the maximum length of a displacement vector, up to approximately the size of the correlation window ϵ (see Fig. 31.15).

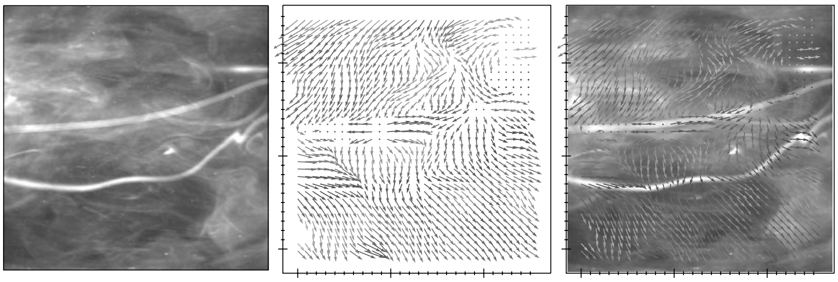


Figure 31.16: *Structure tensor: example of a laser-induced fluorescence (LIF) visualization with the computed displacement vector field.*

The structure tensor approach. Using the 3-D *structure tensor* technique, dense *displacement vector fields* (DVF) can be computed with sub-pixel accuracy on a pixel basis. The approach is based on the detection of linear symmetries and the corresponding orientation angle within a local spatiotemporal neighborhood of *space-time images*. The technique does not depend on the image content and is well suited for particles and for continuous tracers in flow visualization. Section 13.3.2 in Volume 2 gives a detailed description of this method.

This basic property of spatiotemporal images allows estimating the optical flow from a 3-D orientation analysis, searching for the direction of constant gray values in spatiotemporal images. As an example, Figure 31.16 shows the displacement vector field of an LIF image sequence computed with the structure tensor technique. It can be shown that this technique yields displacements with high sub-pixel accuracy of less than 0.01 pixels/frame. However, the maximum displacement is limited by the temporal sampling theorem and can only be extended by the multi-grid technique described in the foregoing section.

31.3.6 Hybrid particle-imaging/particle-tracking velocimetry

Because many particles are within the correlation window of the PIV technique, the resulting velocity field is of lower spatial resolution but is more reliable than the velocity estimate determined by PTV techniques for a single particle. With rather high particle concentrations a high number of correspondences are also lost. Thus it appears to be worthwhile to combine the advantages of PIV and PTV techniques into a hybrid PIV/PTV method.

The simplest approach is to use the velocity field obtained by particle image velocimetry as an initial velocity estimator for the tracking algorithm. Indeed, this combination results in several improvements. First, the length of the extracted trajectories is increased indicating

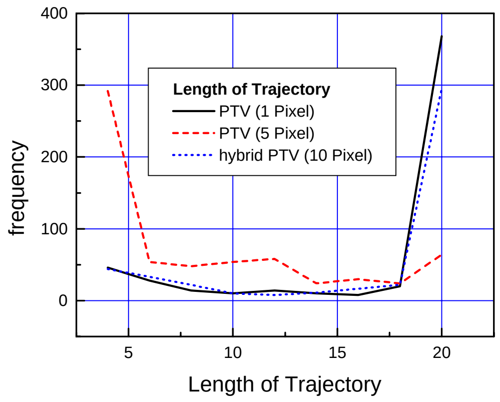


Figure 31.17: *Length of trajectories of particles attached to a rotating disk.*

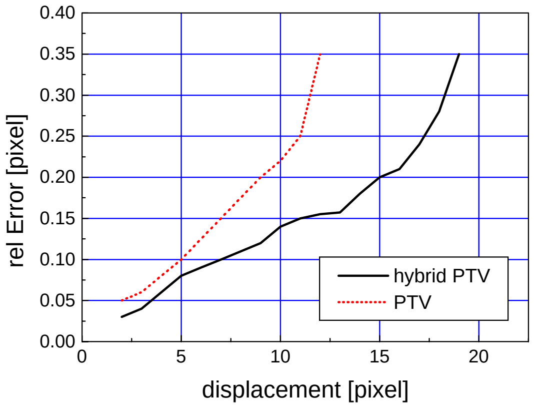


Figure 31.18: *Mean position error for hybrid PTV system.*

that a trajectory is less frequently cut off due to a missing correspondence (Fig. 31.17). Whereas without an initial velocity estimation many particles are lost, the hybrid system allows most of the particles to be tracked even at high displacements. Second, Fig. 31.18 shows that streaks can be tracked over longer image sequences. Third, the relative error in the displacement estimate is significantly reduced, especially for large displacements.

31.4 Stereo particle-tracking velocimetry

The fundamental problem of stereo PTV is the correspondence problem. The stereo correspondence problem is generally solved by comparing images taken from different camera perspectives. This approach

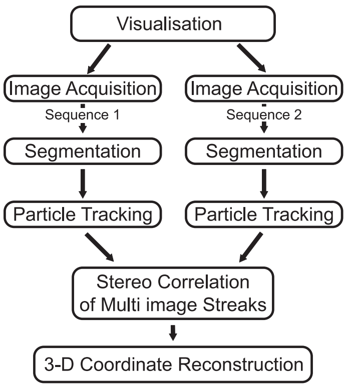


Figure 31.19: *Steps of the stereo PTV algorithm.*

presents ambiguities that become more severe with increasing particle density [25]. There are two ways to solve the correspondence problem. The first way is to solve it by capturing individual images. As shown by Maas et al. [25] this is not possible with only two cameras. The basic problem is that the particles can barely be distinguished from each other. With two cameras, corresponding particles cannot be identified in a unique way since the particle in one image can have a corresponding particle at a line across the whole field of view in the second camera (*epipolar line*, see Volume 2, Chapter 17). Thus, at least three cameras are required for a unique correspondence (two epipolar lines intersect each other in one point if the corresponding stereo bases are not parallel to each other).

The second approach [9] computes the trajectories for each of the stereo cameras first with the PTV techniques described in Section 31.3. The hope, then, is that the trajectories are sufficiently different from each other so that a unique correspondence is possible. The resulting trajectories from each camera perspective can then be matched

The main steps of this algorithm are shown in Fig. 31.19 and will be described in the following sections. The stereo calibration supplies the parameter set for the camera model (Section 31.4.1) used for the stereoscopic reconstruction. The image sequences of the two cameras are then processed separately to extract the trajectories of particles in two dimensions. Finally, the stereo correlation of the trajectories from the two cameras provide the trajectories in 3-D space. In conjunction with the parameter set of the camera model, the spatial coordinates can be obtained.

31.4.1 Geometric camera calibration

To obtain quantitative results in digital image processing the geometric camera calibration is an essential part of the measuring procedure and evaluation. Its purpose is to give a relation between the 3-D world coordinate of an object and its 2-D image coordinates. A geometric camera model describes the projection transformation of a 3-D object onto the 2-D CCD sensor of the camera. The quality of the camera calibration is crucial for the accuracy of the stereoscopic reconstruction. Effects not included in the camera model will result in a systematic error.

31.4.2 The camera model

The camera model $\boldsymbol{F}$ is the mathematical description of the relation between the 3-D world coordinates $\boldsymbol{X}$ and the 2-D image coordinates $\boldsymbol{x}$ (see Volume 1, Chapter 17):

$$\boldsymbol{x} = \boldsymbol{F}(\boldsymbol{X}) \tag{31.22}$$

The simplest model is the pinhole camera model. This model is described by four linear transformations including the ten parameters of translation $\boldsymbol{T}$ (3 offsets), rotation $\boldsymbol{M}$ (3 angles), projection $\boldsymbol{P}$ (focal point) and scaling S

$$\boldsymbol{F} = S\boldsymbol{T}\boldsymbol{M}\boldsymbol{P} \tag{31.23}$$

An extended model includes also lens distortion (compare Volume 1, Section 4.5.5 and Chapter 17). Lens distortion is described by a nonlinear correction (see Hartley [26]).

Multimedia geometry. In many applications the camera and the optical system have to be mounted outside the volume of observation. The camera looks, for example, through a glass or Perspex window into the water (see Fig. 31.20). Therefore, a multimedia correction seems to be necessary. The light is refracted according to Snellius' law. Its effect is described by a displacement from the straight line $\Delta X = X_0 - X_{mm}$. If the geometry of the setup is known, ΔX is a function of X_w and of the location of the camera X_k. As this function has no analytical solution, ΔX has to be calculated numerically applying Fermat's principle.

Parameter estimation. The nonlinearity of the camera model does not allow computing the parameters in a direct analytical way. Therefore, a nonlinear minimization method (Levenberg-Marquardt algorithm [18]) is used. The function to minimize is

$$\mathrm{Res} = \sum_i (x_i - \mathbf{F}(X_i))^2 \tag{31.24}$$

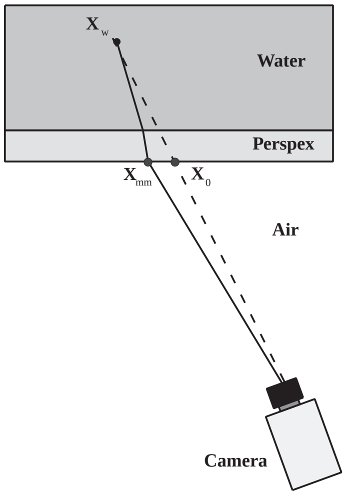


Figure 31.20: *Distortion by multimedia geometry.*

To avoid the algorithm from getting caught in a local minimum, a choice of proper initial values is critical. This can be guaranteed by neglecting the nonlinear part of the camera model (k_1, k_2, p_1, and p_2 set to zero). The parameters of the remaining linear model which are determined by a direct linear transform (DLT) (see Melen [27]), yield the initial values.

Inversion of the camera model. The camera model describes the transformation from world to image coordinates. For the evaluation the inverse relation is of interest—to obtain world coordinates of an object from its position in the 2-D image.

Because the camera model projects a point in 3-D space to a point in a 2-D plane, the solution of the inversion of the camera model is a line. The intersection of the two lines gives the 3-D position of the object (triangulation problem [26]). Mainly due to noise error the lines do not exactly intersect. Therefore, the midpoint of the minimum distance of the two lines is taken as the "intersection point." To invert the camera model a numerical method has to be used because an analytical solution does not exist. For an image point $\boldsymbol{x}$ and a given Z-coordinate Z_0 the world point $\boldsymbol{X}$ to be found is given by the minimum of

$$\epsilon = \left| \boldsymbol{x} - \mathbf{F}(\boldsymbol{X}) \right|_{Z=Z_0} \qquad (31.25)$$

As this function is convex, a gradient-based minimization routine always converges.

Table 31.1: *Comparison of camera models*

Camera model	Residue in pixel^2
Without multimedia correction	0.121
With multimedia correction	0.325

31.4.3 Virtual camera

First, the accuracy of the camera model can be tested with and without the multimedia correction. The quality of the camera model can be quantified by the residue as defined in Eq. (31.25).

As shown in Table 31.1, the model without the multimedia correction yields lower residue and therefore a better result. Taking multimedia correction into account, the exact position of the two cameras is required. In the camera model the z coordinate of the camera and the focal length are highly correlated parameters. Therefore, the camera position is quite uncertain and the multimedia correction is not exact.

Because of the lower residue the model that does not take multimedia correction into account is better suited. Therefore the positions of the cameras are virtual. The actual world coordinates of the cameras are not necessary for the stereo reconstruction described in the following sections.

31.4.4 Stereoscopic correspondence

Using a calibrated stereo rig the 3-D trajectories can be reconstructed from the 2-D trajectories by solving the stereoscopic correspondence problem. In order to reconstruct the location of the trajectory in 3-D space, its image coordinates (x, y) and (x', y') for both cameras have to be found. This will be described in detail in the following sections. A general discussion on the geometry of stereo imaging and the determination of the stereo disparity can be found in Volume 2, Chapters 17 and 18, respectively. From (x, y) and (x', y') it is possible to perform the triangulation with the knowledge gained by the calibration procedure discussed in Section 31.4.1.

The epipolar constraint. The computational efficiency of establishing stereoscopic correspondences is increased by introducing the *epipolar constraint*. It is derived from stereoscopic geometry and as such is a very strong constraint. As stated in Section 31.4.2, the object has to be located on the outgoing ray of the first camera as can be seen in Fig. 31.21. This ray is in turn depicted by the second camera as the so-called *epipolar line*. The endpoints of the epipolar line are called

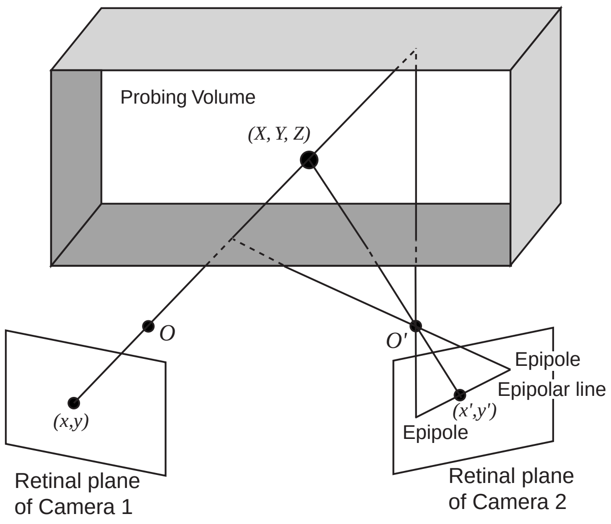


Figure 31.21: *Construction of the epipolar line in the retinal plane of camera 2 from an image point* (x, y) *on the retinal plane of camera 1.*

epipoles. They are defined by the finite depth range of the probing volume. This constraint is, of course, symmetric for both cameras.

Due to noise and a small error in determining the camera parameters a certain tolerance ϵ is added to the epipolar line. It will thus be a narrow window in image space as is shown in Fig. 31.22.

Taking lens distortion and multimedia geometry into account, the epipolar line will deviate from a straight line and be slightly bent. In practice this curvature proved to be small enough to be accounted for by the tolerance ϵ. For practical purposes the epipolar line is thus replaced by an epipolar window.

31.4.5 The stereo correlation algorithm

In a first step a list of possibly corresponding candidates is constructed. The candidates obey two constraints:

- the time constraint: corresponding trajectories were tracked at the same time; and
- the *epipolar constraint*: the corresponding trajectory T_2 to trajectory T_1 was found within the epipolar window.

Depending on particle densities these two constraints may suffice to allow a unique solution of the correspondence problem. This may not always be the case as high particle densities are beneficial for high

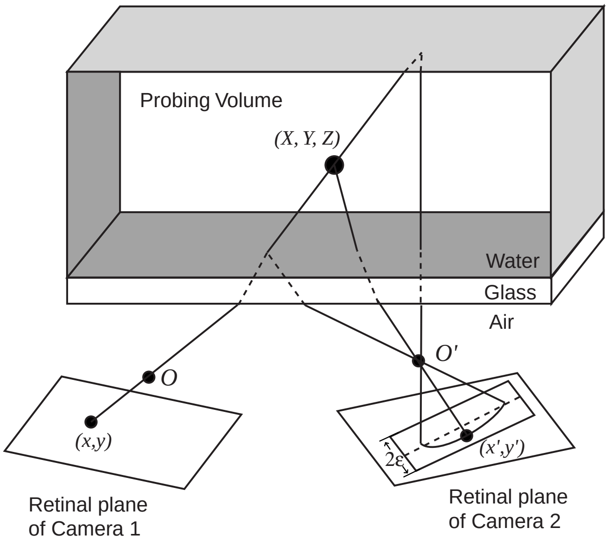


Figure 31.22: *The epipolar line for a multimedia setup. The tolerance ϵ is added to compensate for noise and the curvature.*

spatial resolutions in visualizing flows.

Generally, there are four possible outcomes of the correspondence search (see Fig. 31.23):

(1) *no correspondence:* To a trajectory T_1 in the first camera no corresponding trajectory T_2 was found in the second image;

(2) *one-to-one correspondence:* Exactly one corresponding trajectory T_2 was found for a trajectory T_1;

(3) *one-to-many correspondence:* For a trajectory T_1 multiple trajectories $T_{2,i}$ were found, which, in turn all correspond to the same trajectory T_1; and

(4) *many-to-many correspondence:* Multiple trajectories $T_{2,i}$ were found for a trajectory T_1, whereby $T_{2,i}$ also corresponds to different trajectories $T_{1,i}$.

In case (1) there exist no corresponding pairs of trajectories for which the triangulation could be performed, thus no world coordinates (X, Y, Z) can be computed. This occurs when a particle is imaged by only one camera. To avoid this it is important to maximize the overlap of the two camera footprints. Another reason might be that streaks are not segmented and tracked correctly. As a countermeasure the illumi-

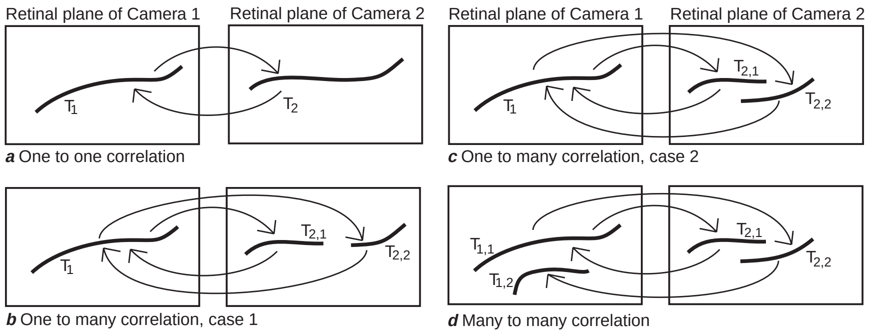

Retinal plane of Camera 1
Retinal plane of Camera 2
T1
T2
a One to one correlation
T2,1
T2,2
c One to many correlation, case 2
b One to many correlation, case 1
T1,1
T1,2
d Many to many correlation

***Figure 31.23:** Some possible cases that may occur during correspondence search.*

nation should be homogeneous over the whole visible volume of both cameras.

For case (2) the correlation was established and the world coordinates (X, Y, Z) of the trajectory can readily be calculated.

There are two distinct possibilities for case (3) to occur. The first is that the trajectories $T_{2,i}$ do not overlap in the time domain. This may be the case when a streak in the second camera was not segmented correctly over the whole time interval of the trajectory T_1. The resulting trajectories $T_{2,i}$ are thus generated by the same particle and can be marked as corresponding to T_1.

Case (3) occurs if the trajectories $T_{2,i}$ overlap at the same point in time. Clearly, this violates the boundary condition that trajectories have to be unique. The trajectories $T_{2,i}$ must thus belong to different particles. New criteria have to be found to overcome these ambiguities. This possibility is thus akin to case (4), where multiple trajectories $T_{2,i}$ were found for a trajectory T_1, all of which correspond to different trajectories $T_{1,i}$.

Criteria for resolving ambiguities. A strong constraint in solving the remaining ambiguities is the uniqueness constraint. This constraint states that an object point projected onto the first image should match at most with just one image point in the other view. This constraint does not hold for transparent objects over a certain size such as bubbles, where reflections may result in a single image point in one view and in two image points in the other.

For flow visualization the use of a symmetric setup with respect to illumination of small particles that are rotationally symmetric is of advantage. The particles may thus be viewed as point objects to which the uniqueness constraint may be applied.

For experiments at the bubble column—where the diameter of the bubbles is typically up to 0.5 cm—it is, of course, desirable to make use of this constraint as well. This is achieved by positioning the two cameras in such a way that their retinal planes enclose a small angle of 20° to 30° only. The bubble can then be viewed as a single point in space.

The trajectories are constructed from individually segmented streaks S_i. Invalid segmentation may therefore lead to limitations of the uniqueness constraint as not all streaks $S_{1,i}$ in the first image correlate to streaks $S_{2,j}$ in the second image. This is accounted for by a heuristic threshold value K that is determined by the number of correlated streaks k of the trajectories to noncorrelated streaks n, or $K = k/n$. If the trajectory T_1 comprises i streaks $S_{1,i}$ and the trajectory T_2 comprises j streaks $S_{2,j}$, then the number of correlated streaks k is given by $k = \min(j,i)$ and the number of uncorrelated streaks n is given by $n = |j - i|$. Therefore the threshold K can be written as

$$K = k/n = \min(j,i)/|j - i| \tag{31.26}$$

Another possible solution to the correspondence problem is statistical in nature. We stated in the foregoing that in real-world situations correlated streaks in one image will generally not be found exactly on the epipolar line of the second image due to noise and camera distortion. Therefore, the standard deviation σ_d can be calculated for the distance d from all the streaks of correlated trajectories to the epipolar line.

If the two trajectories correspond to each other, the distance d will be due solely to the tolerance of calculating the epipolar line. It should roughly be the same for all streaks of the trajectory and accordingly the standard deviation σ_d remains small.

If the two trajectories do not correspond, then d will be proportional to the relative motion of the two tracked particles and σ_d is expected to be much greater than in the other case. This makes it possible to solve some ambiguities by thresholding σ_d.

31.4.6 Results

In Fig. 31.24 the trajectories of the uprising bubbles can be seen. On the left part of the image the bubbles move upwards in a straight motion, on the right part of the image the flow is more turbulent (see Chen et al. [28]). The sequence consists of 350 images, and about 300 3-D trajectories were extracted from about 3000 trajectories obtained by PTV.

Figure 31.25 shows the trajectories of visualized and tracked particles, representing the flow beneath a free wind-driven wavy water surface. The sequence of 500 images covers a time interval of 8.3 s. The

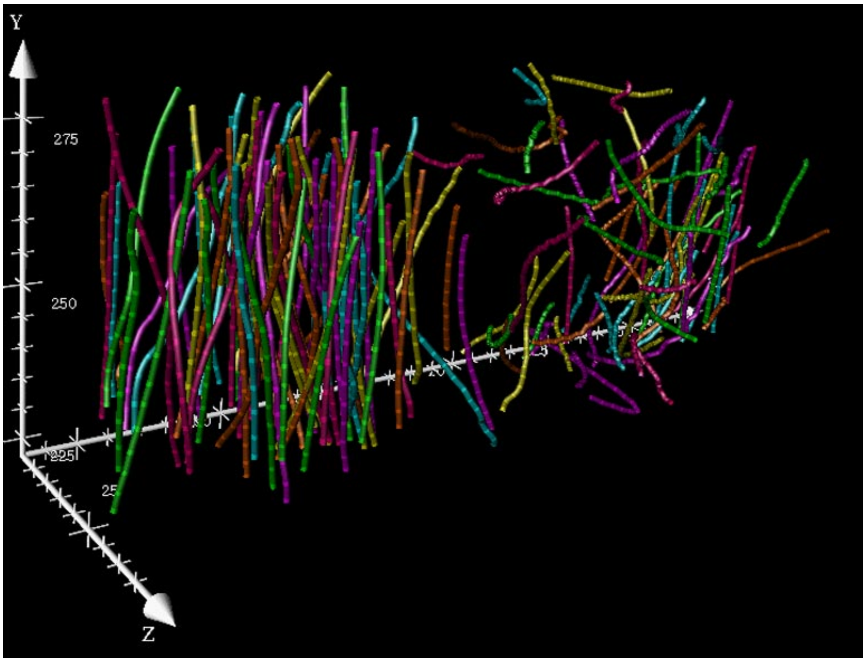


***Figure 31.24:** Three-dimensional trajectories of bubbles from bubble column.*

spirally shaped trajectories result from the orbital motion of small-amplitude (< 0.5 cm) gravity waves with little steepness ($< 5°$).

Depending on conditions, from about 10 to 30 % of the 2-D trajectories in the stereo images, 3-D trajectories can be computed. The smaller number of 3-D trajectories compared to the 2-D trajectories is also due to geometric reduction of the intersecting volume from the two cameras. For example, if the angle α enclosed by the two cameras (Fig. 31.5) is 60° and the depth of focus is 4 cm, the maximum intersecting volume is 75 % (ideal case) of the volume observed by a single camera. The large α is chosen in favor of a higher resolution in depth (z-direction). If the light intensity is homogeneous and constant over time, the PTV can track particle or bubble trajectories over the whole sequence of images. This is well illustrated by the results gained from the bubble column in Fig. 31.2.

In the instance of flow visualization in the wind-wave flume the situation is more difficult. The number of reconstructed 3-D trajectories is much less than extracted by PTV and the length of the trajectories is shorter compared to the bubble column. Therefore, the higher density, the smaller size, and the greater velocity of the particles increase the number of ambiguities in the correlation step. If the variation of steepness is small between successive images, which is the case for

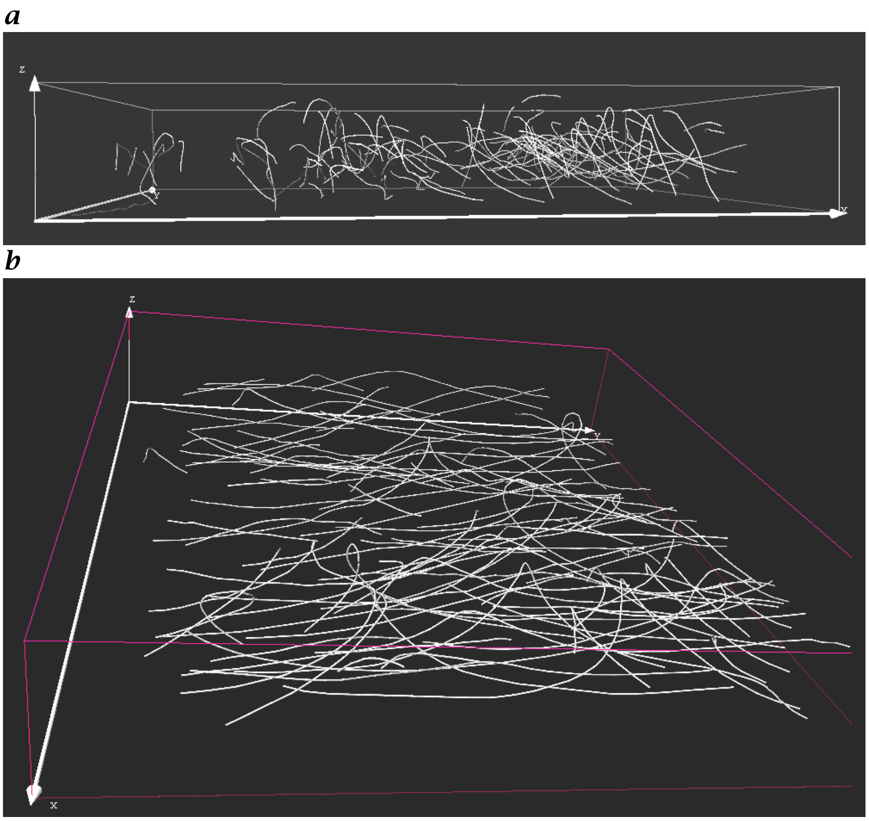


Figure 31.25: Three-dimensional trajectories visualizing flow, two different perspectives: ***a*** *perpendicular to the water surface and parallel to the main direction of flow;* ***b*** *perpendicular to the main direction of flow and in a slight angle onto the water surface.*

small-amplitude gravitational waves, the PTV works well to gain an insight into the 3-D flow field pattern beneath the water surface.

31.5 Conclusions

Stereo particle-tracking velocimetry is a powerful method for studying dynamic processes. For applications in flow field visualization the tracking of particles works for a concentration of up to 800 particles per image. In a 512×512 image this corresponds to a mean distance between the particles of about 30 pixels. Table 31.2 shows a comparison of the previously described flow field diagnostic techniques. For comparison all numbers given refer to an image size of 512×512 pixels.

***Table 31.2:** Comparison of particle-tracking velocimetry methods*

	PIV	Hybrid PTV	Stereo PTV	Structure tensor
Tracer	discrete	discrete	discrete	continuous
Flow field	DVF	DVF *and* trajectories	trajectories	DVF
Particle Density	≈ 3000	< 1000	< 1000	–
Averaging	spatial	spatial (DFV)	no	spatial, temporal
Measure of certainty	yes	no	no	yes

Investigations of flow dynamics close to the air-water interface in a wind-wave flume and in a bubble column were shown as sample applications of stereo PTV. The method is useful for many other purposes in addition to flow visual applications. The only requirements for visualization are visible tracer objects such as seeding particles.

With stereo PTV a high spatial and temporal resolution can be obtained. Applying the calibration method described in Section 31.4.1 the spatial resolution can be sub-pixel accurate. The temporal resolution depends on the image acquisition rate of the CCD camera and on the frame grabber alone. PTV yields both the Eulerian and Lagrangian representation of the flow field, whereas PIV gives the Eulerian representation only. The Eulerian representation can be calculated from the Lagrangian representation. This clearly shows the advantage of the presented PTV methods in the study of dynamic processes. Nevertheless, the density of tracer particles in PIV methods can be greater (up to a few thousand/image) than for PTV methods. The combination of both techniques such as the hybrid PIV/PTV described in Section 31.3.6 can produce even better results.

For practical purposes an easy experimental setup is always of advantage. Therefore, the simple stereo PTV setup described in section Section 31.2.1 and Section 31.4, including only two cameras, is a useful choice to determine 3-D trajectories in a measuring volume.

Acknowledgments

Financial support for this research from the U.S. National Science Foundation through grant OCE91-15994 and the German Science Foundation (DFG) through grant Wi 1029/2-1 is gratefully acknowledged.

31.6 References

[1] Koochesfahani, M. M. and Dimotakis, P. E., (1985). Laser induced fluorescence measurements of mixed fluid concentration in a liquid shear layer. *AiAA Journ.*, **23**:1699–1707.

[2] Koochesfahani, M. M. and Dimotakis, P. E., (1986). Mixing and chemical reactions in a turbulent liquid mixing layer. *J. Fluid Mech.*, **170**:83–112.

[3] Maas, H.-G., Stefanidis, A., and Grün, A., (1994). From pixels to voxels—tracking volume elements in sequences of 3-D digital images. In *IAPRS Proc.*, Vol. 30, p. Part 3/2.

[4] Grant, I., (1997). Particle image velocimetry: a review. In *Proc. Instn. Mech. Engrs.*, Vol. 211 of *C*, pp. 55–76.

[5] Adrian, R., (1991). Particle-imaging techniques for experimental fluid mechanics. *Ann. Rev. Fluid Mech.*, **23**:261–304.

[6] Hesselink, L., (1988). Digital image processing in flow visualization. *Ann. Rev. Fluid Mech*, **20**:421–485.

[7] Gharib, M. and Willert, C., (1988). Particle tracing revisited. In *Proceedings, 1st National Fluid Dynamics Congress, Cincinnati/Ohio, AIAA-88-3776-CP.*

[8] Adamczyk, A. and Rimai, L., (1988). 2-Dimensional particle tracking velocimetry (PTV): Technique and image processing algorithm. *Experiments in Fluids*, **6**:373–380.

[9] Netzsch, T. and Jähne, B., (1995). A high performance for 3-dimensional particle tracking velocimetry in turbulent flow research using image sequences. *Proc. ISPRS Intercommission Workshop 'From Pixels to Sequences', Zurich*, **30**(5W1).

[10] Schmitt, F. and Ruck, B., (1986). Laserlichtschnittverfahren zur qualitativen Strömungsanalyse. *Laser und Optoelektronik*, **2**:107–118.

[11] Hinze, J., (1959). *Turbulence.* New York: McGraw-Hill.

[12] Wierzimok, D. and Jähne, B., (1991). Measurement of wave-induced turbulent flow structures using image sequence analysis. In *2nd Int. Symp. Gas Transfer at Water Surfaces, Minneapolis 1990.* Am. Soc. Civ. Eng.

[13] Hering, F., Leue, C., Wierzimok, D., and Jähne, B., (1997). Particle tracking velocimetry beneath water waves part I: Visualization and tracking algorithms. *Experiments in Fluids*, **23**(6):472–482.

[14] Hering, F., Leue, C., Wierzimok, D., and Jähne, B., (1998). Particle tracking velocimetry beneath water waves part II: water waves. *Experiments in Fluids*, **24**:10–16.

[15] Hering, F., (1996). *Lagrangesche Untersuchungen des Strömungsfeldes unterhalb der wellenbewegten Wasseroberfläche mittels Bildfolgenanalyse.* , Institute for Environmental Physics, University of Heidelberg.

[16] Leue, C., Hering, F., Geissler, P., and Jähne, B., (1996). Segmentierung von Partikelbildern in der Strömungsvisualisierung. In *Proc. 18. DAGM-Symposium Mustererkennung, Informatik Aktuell*, B. Springer, ed., pp. 118–129.

[17] Jähne, B., (1996). *Digital Image Processing: Concepts, Algorithms and Scientific Applications.* 2nd edition. Berlin: Springer.

[18] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., (1992). *Numerical Recipes in C: The Art of Scientific Computing.* New York: Cambridge University Press.

[19] Wierzimok, D. and Hering, F., (1993). Quantitative imaging of transport in fluids with digital particle tracking velocimetry. *Imaging in Transport Processes, Begell House*, pp. 297–308.

[20] Hering, F., Merle, M., Wierzimok, D., and Jähne, B., (1995). A robust technique for tracking particles over long image sequences. In *Proc. ISPRS Intercommission Workshop 'From Pixels to Sequences,' Zurich, In Int'l Arch. of Photog. and Rem. Sens.*, Vol. 30 of *5W1*.

[21] Etoh, T. and Takehara, K., (1992). An algorithm for tracking particles. In *Proceedings, 6th International Symposium on Stochastic Hydraulics, Tapei.*

[22] Agüí, J. and Jiménez, J., (1987). On the performance of particle tracking. *J. Fluid Mech.*, **185**:447–468.

[23] Spedding, G. and Rignot, (1990). Performance analysis of grid interpolation techniques for fluid flows. *Exp. Fluids*, **15**:417–430.

[24] Burt, P. and Adelson, E., (1983). The Laplacian pyramid as a compact image code. *IEEE Trans. COMM.*, **31**:532–540.

[25] Maas, H. G., Gruen, A., and Papantoniou, D., (1992). Particle tracking velocimetry in three-dimensional flows. Part 1: Photogrammetric determination of particle coordinates. *Exp. Fluids*, **15**:133–146.

[26] Hartley, R., (1997). Triangulation. *Computer Vision and Image Understanding*, **68**(2):146–157.

[27] Melen, T., (1994). *Geometrical Modeling and Calibration of Video Cameras for Underwater Navigation.* PhD thesis, Norges tekniske høgskole, Institutt for teknisk kybernetikk, Trondheim, Norway.

[28] Chen, R., Reese, J., and Fan, L.-S., (1994). Flow structure in a three-dimensional bubble column and three-phase fluidized bed. *AIChE J.*, **40**: 1093.

32 Analyzing Particle Movements at Soil Interfaces

Hagen Spies[1,2], Oliver Beringer[1], Hermann Gröning[1], and Horst Haußecker[1]

[1] Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR) Universität Heidelberg, Germany
[2] Dept. of Computer Science, University of Western Ontario, Canada

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

32.1 Introduction

In order to protect utilizable floor space from rivers changing their course it has long been necessary to fortify embankments. The slope stability of embankment constructions is greatly influenced by *pore water pressures* induced by hydraulic load changes as a result of ship traffic. This influence was analyzed in laboratory measurements that are presented here. Air bubbles contained in the pore water greatly delay pressure transmission. Thus changing hydrodynamic load conditions may lead to locally and temporally acting excess pore water pressure and therefore to transient *pore water flow*. The transport of movable soil particles is strongly dependent on the acting *hydraulic gradients* in the water flow. This water flow may in turn cause destabilization of the soil structure. Under certain conditions even *fluidization* of soil layers occurs, resulting in the collapse of the embankment.

32.2 Previous investigations

Previous investigations focused on the pressure distribution within soil structures. From pressure gauges inserted at various locations it could be inferred that pressure changes do not spread unhindered. From empirical investigations the following formula for the *excess pore water* pressure Δp depending on the depth h was derived [1]

$$\Delta p(h,t) = \rho_w \, \Delta h \, g \, (1 - a(t)\, e^{-b(t)h}) \tag{32.1}$$

where ρ_w is the density of water, Δh is the water level change, and g is the gravity acceleration. The two time-dependent parameters $a(t)$ and $b(t)$ vary with the soil type and the acting hydraulic load. The pressure distribution is governed mainly by $b(t)$[1/m], which in turn depends on the permeability of the soil and the hydraulic gradient. However, no attempt has been made to study the processes on a microscopic scale. The following sections describe our approach to the task that is based on the analysis of images gathered with endoscopes.

At a previous stage of the project we focused on the detection of moving areas within the soil structure [2, 3, 4]. From those areas, motion frequency can be deduced and compared to the amount of motion expected at the given pressure situation. These experiments showed promising results, thus, we extended our image analysis towards a quantitative investigation of sediment motion.

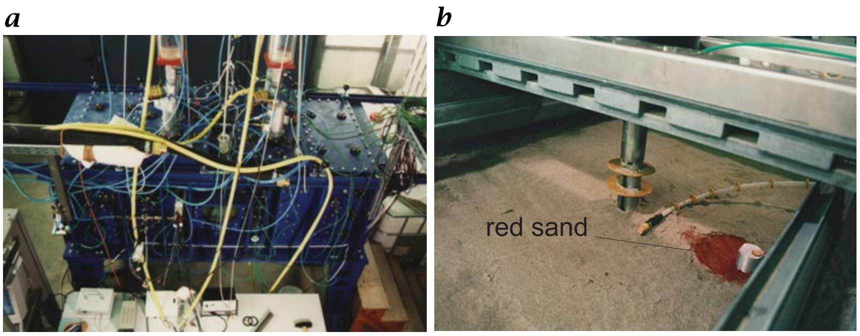


Figure 32.1: ***a*** *The pressure tank during measurements;* ***b*** *mounting of endoscopes and pressure gauges.*

32.3 Experimental setup

Experiments where carried out at a specifically constructed large pressure tank located at the Federal Waterways Engineering and Research Institute in Karlsruhe, Germany.

32.3.1 The pressure tank

The tank itself is roughly 2 m long, 80 cm wide and 80 cm high. With this unique device it is possible to apply pressure gradients in all directions superimposed by steady and transient water flow in a pressure environment of up to 3 bar. Pressure loadings may act from above and underneath the soil structure simultaneously and the water can be forced to flow either above the *sediment layers* or directly through the sediment in separate depth layers. Several adjustable endoscope inlets allow all critical areas within the sediment to be observed. Figure 32.1a shows the pressure tank during the summer 1997 measurement campaign.

During this campaign, the tank was filled with sand of a diameter range from 0.05 to 0.5 mm. On top of this main sediment body was a 10-cm layer of gravel. This layer is again stabilized from above by a variable hydraulic pressure-distributed via steel bars and metal sheets. This allows simuling revetment or other load that is usually present to stabilize river embankments. It is one goal of our research to study how much additional load is needed to maintain a stable configuration.

At one location directly below the gravel, two layers of red and green colored sand are placed. In Fig. 32.1b, the top of the red layer can be observed. One of the endoscopes is placed in such a way that the interface between the two color layers can be imaged. From these images, it is then possible to obtain information about mixing in that area (see

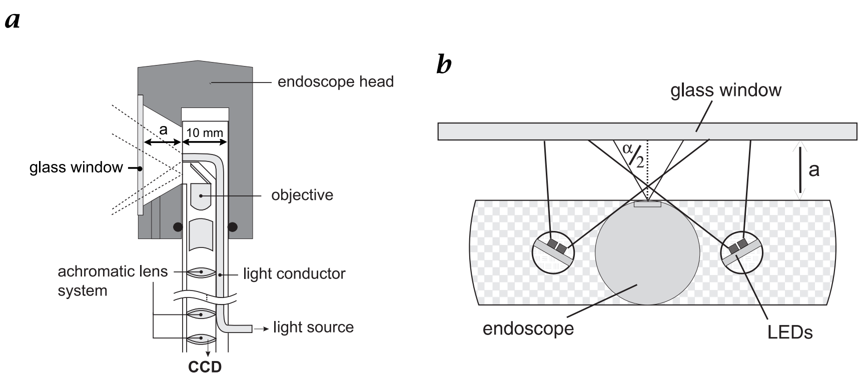


Figure 32.2: *Setup of the endoscope:* ***a*** *vertical cross section through the endoscope; and* ***b*** *horizontal cross section with the new illumination setup using LED's.*

Section 32.4.5). The other endoscopes are placed directly at the gravel-sand interface. Additionally, a flexible endoscope is used within the gravel layer to study the water flow within the pore volume.

32.3.2 Endoscopes

We use rigid *endoscopes* with a viewing direction of 90° and a diameter of 10 mm. An external cold light source coupled via a fiber light conductor to the endoscope can be used for illumination (Fig. 32.2). With a viewing distance of $a = 8$ mm and an aperture angle $\alpha = 60°$ we obtain a viewing area of roughly 6×6 mm. An achromatic lens system allows magnification to be adjusted by a factor of up to 2. For exact geometric calibration we used a grid target. Cross sections through the endoscope and protection head are given in Fig. 32.2a.

Homogeneous illumination using LED's. Using the *fiber optic illumination* integrated in the endoscope, specular reflexes on the glass window can not be avoided. These and the *inhomogeneous illumination* result in over- and underexposed regions that drastically reduce the quality of the acquired images. Hence, a new setup has been devised with *LED* panels on both sides of the endoscopes. Thus, the angle of incidence on the glass window is far smaller than with the fiber light conductor. As a result, no reflexes occur in the viewing area. This setup is shown in Fig. 32.2b. The improvement is illustrated in Fig. 32.3a and b where a paper is placed on the glass window.

Flexible endoscope. The flexible endoscope consists of circularly arranged optic fibers. The resulting bundle has a diameter of 1.1 mm and

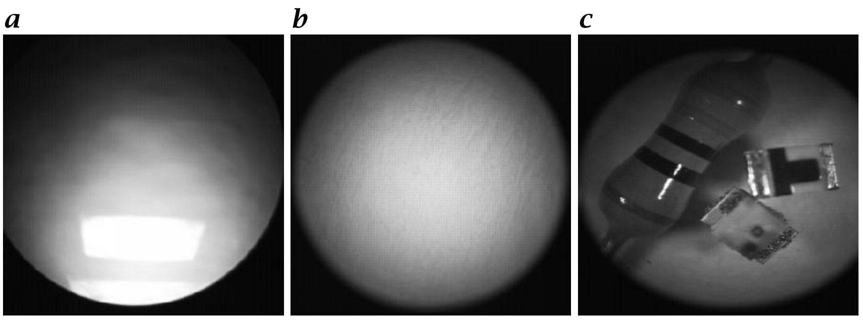


Figure 32.3: Comparison of the new LED-based and old fiber-illumination setup: ***a*** *old setup;* ***b*** *and* ***c*** *new setup.*

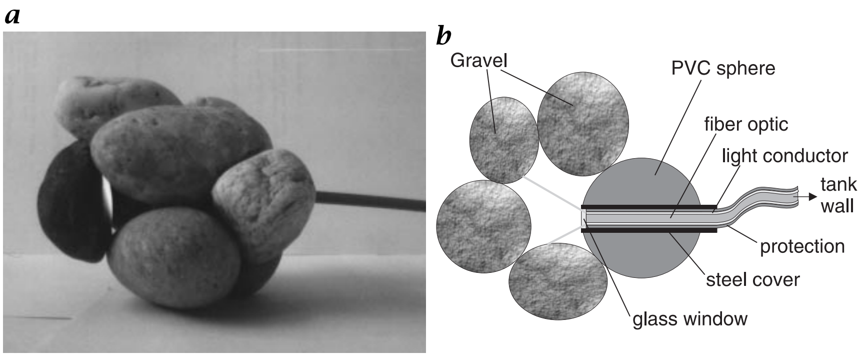


Figure 32.4: The flexible endoscope: ***a*** *placed in a gravel package; and* ***b*** *cross section through the setup.*

a length of 80 cm. For illumination, a cold light source is coupled to a ring of fiber conductors. For utilization in the rough environment of the pressure tank, the actual endoscope is surrounded by a protection cover, increasing the diameter to 1.4 mm. The end is covered by a 2-cm long steel tube with a glass window at the end. In order to introduce as little disturbance to the water current as possible, the endoscope is placed in a PVC sphere of similar size as the surrounding gravel stones to form a pore volume. Figure 32.4 shows the described setup.

32.3.3 Flow visualization

For visualization of the water flow latex particles are added. These particles possess nearly the same density as water, which makes them well suited for flow analysis. (For a detailed discussion of quantitative flow measurements via particle tracking techniques see Chapter 31).

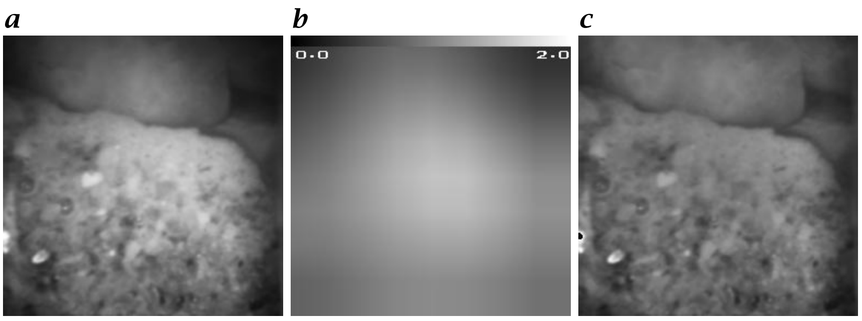


Figure 32.5: Illumination correction: ***a*** *example image;* ***b*** *smoothed image normalized to a mean of 1; and* ***c*** *corrected image.*

32.4 Image analysis

32.4.1 Correcting for inhomogeneous illumination

Due to two different causes the collected imagery is brighter in the center:

- First, when using the fiber optic light conductor, the illumination is not homogeneous and often specular reflexes occur. With the new LED illumination setup as described in Section 32.3.2, these reflexes can be avoided and a far better illumination is achieved.
- The second effect is due to vignetting that results from the wide angle lens system integrated in the endoscope. Vignetting can not be avoided with the present setup and correction has to be made for it. For correction we use a normalized, highly smoothed image as correction factor. The result can be seen in Fig. 32.5 where the fifth level of a Gaussian pyramid is used for smoothing of the example image.

32.4.2 Motion detection

In the first phase of the investigations, grain movements were detected. Relating the amount of movement to the acting hydraulic loads gives insights into the conditions under which the soil structure becomes unstable. As a wide range from very slow to rapid motion is encountered, a special *motion detection* algorithm is needed that is introduced next.

We do not intend to detect the complete area of the moving particles. It is only the boundaries of low textured objects that appear to move. Coding only these pixels provides enough information for reconstruction. Thus, we use a simplified motion-detection algorithm for compression purposes. For noise reduction and efficiency reasons, we use the first level of a Gaussian pyramid. A special implementation of the Gaussian pyramid significantly reduces the number of computations [5]. Second, a binomial smoothing operator is applied in the time direction. The resulting images show practically no noise and changing areas are easily detected using a threshold on the temporal derivative. This threshold is set rather low and outliers are removed by a subsequent morphological erosion.

32.4.3 Motion compression

The detection of moving areas within image sequences can be readily exploited for *image compression* purposes. The scenes under consideration often show long periods of little or no motion. Storing the complete images thus introduces considerable redundancy. We typically investigate 800 images of size 512×512 pixels, resulting in approximately 205 MB of data. Using European video norm (PAL, 25 frames/s), this equates to 32 s of observation time. For colored images, this number has to be halved again as both red and green channels are used. During a measurement campaign, large amounts of data quickly accumulate. Lossless compression algorithms such as *LZW* or *ZIP* achieve reduction rates of approximately 20 % on our data. In order to achieve better compression rates we used a lossy method described in this section.

The goal of the envisaged compression was not to be lossless with regard to gray-value information but rather to keep the derived physical results unchanged. The fundamental image-processing task considered here is the estimation of velocities. We, therefore, examined to which extent the calculation of optical flow is influenced by our compression technique.

For compression we proceed as follows: From the original image sequence we compute the moving areas, and the remaining pixels are set to zero. The resulting image contains large black areas and is well suited for standard compression techniques. We achieved compression rates of more than 80 %. Reconstruction is done by recursively using previously reconstructed pixels where no motion was detected, so that only one image is needed for the initialization of the reconstruction algorithm. Figure 32.6 shows a flowchart of the algorithm.

Figure 32.7 shows the first image of three generated test sequences that were used for evaluating the algorithm. In all three test cases, the objects were fully detected for velocities exceeding 0.05 pixels/frame. Smaller velocities lead to only partially segmented objects. The gray-level difference between original and reconstructed image remains on average below two. For the test, normal distributed noise with a standard deviation of 1.2 gray levels was added to the images, corresponding to the typical noise levels as found for our cameras.

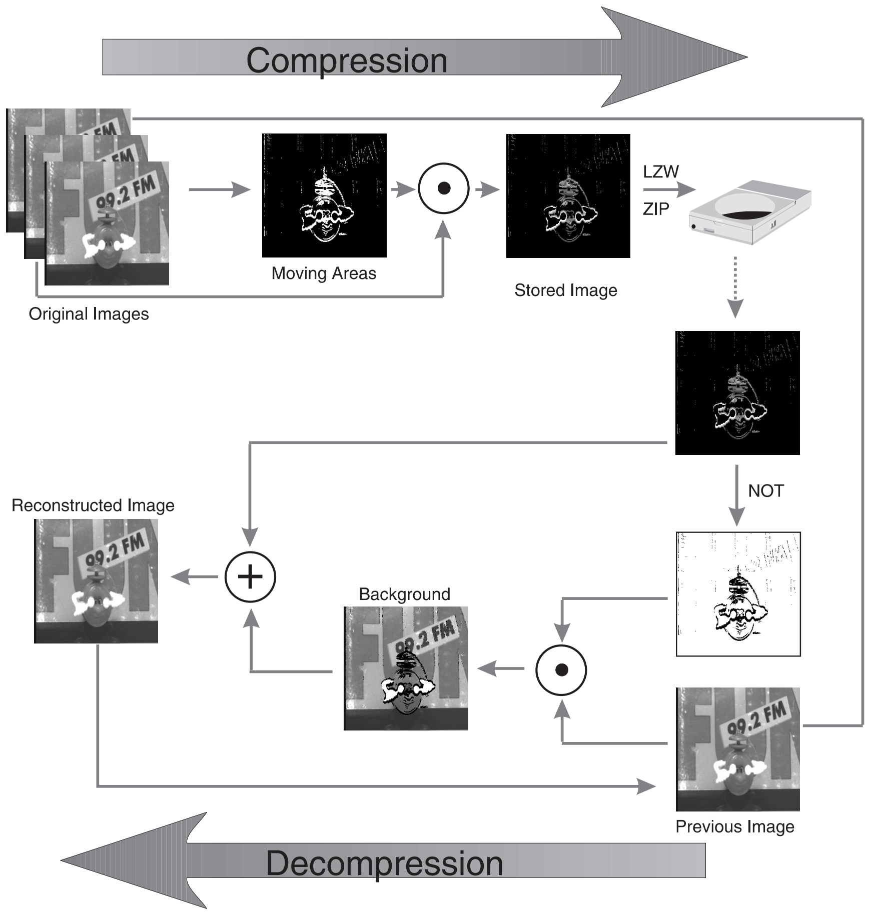


Figure 32.6: The steps involved in the compression algorithm.

As mentioned in the foregoing, the idea behind the compression was not to change the final velocity estimates. An investigation towards this end is presented in the next section.

32.4.4 Optical flow estimation

For velocity calculation of soil movements, we use the structure-tensor technique as described in Volume 2, Section 13.3.2. As the image material used here is relatively low in contrast, a low threshold on the trace of the tensor has to be used. For the results described in the following, we used only areas where the trace exceeded a value of 5. Furthermore, there are areas that show very little spatial variation that makes motion estimation difficult. The coherence measure provides a criterion to de-

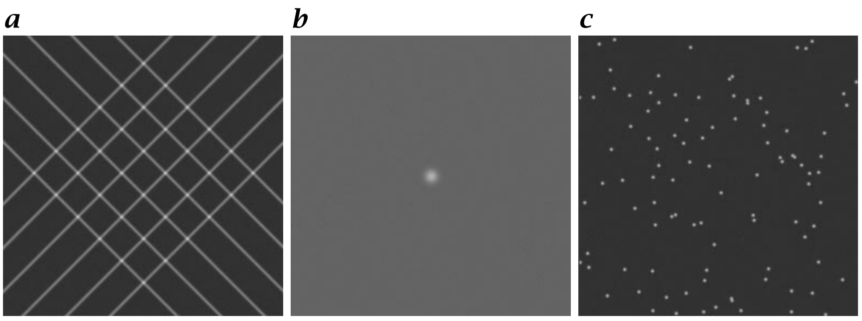


Figure 32.7: One frame of the generated test images used: ***a*** *grid;* ***b*** *single Gaussian ($\sigma = 5\,pixel$); and* ***c*** *multiple Gaussians ($\sigma = 1\,pixel$).*

cide how reliable the *optical flow* computation was performed. With the given data, it is necessary to use a rather low threshold of 0.5 on the coherence. Otherwise, the displacement vector field becomes to sparse.

A survey on the achievable accuracy with the tensor method under these circumstances showed a strong dependence on image content. Figure 32.8a shows the relative error in velocity estimation on the generated test data of Fig. 32.7b and c with single and multiple Gaussians. Due to the small object size, aliasing effects deteriorate the achieved accuracy dramatically for velocities over 1 pixel/frame. Second, those Gaussians pose a difficult problem also at small velocities because they contain large regions with very low contrast. Images of soil layers typically show more contrast, larger objects and moderate velocities. For the typical velocity range encountered here (0.05 to 1 pixel/frame), we found that a relative error of 5 % can be used as an upper margin.

Compression and optical flow calculation. As stated before, the main objective for our compression is not to hinder the accurate estimation of optical flow. We used the generated test data of Fig. 32.7b and c to investigate the influence of compression and decompression prior to the analysis with the structure tensor. As can be seen in Fig. 32.8a, the compression does not alter the performance. Figure 32.8b shows the computed optical flow on an example sequence of the gravel-sand boundary. Again, the compression does not alter the outcome.

Normalized convolution. As described in the foregoing, the computation of optical flow can only be performed accurately on the subset of the image where the coherence exceeds a certain threshold. Even though we use a low value, the resulting displacement field is rather sparse (Fig. 32.9a). We assume the movement of the soil structure to be smooth due to frictional forces between particles. Then, a dense ve-

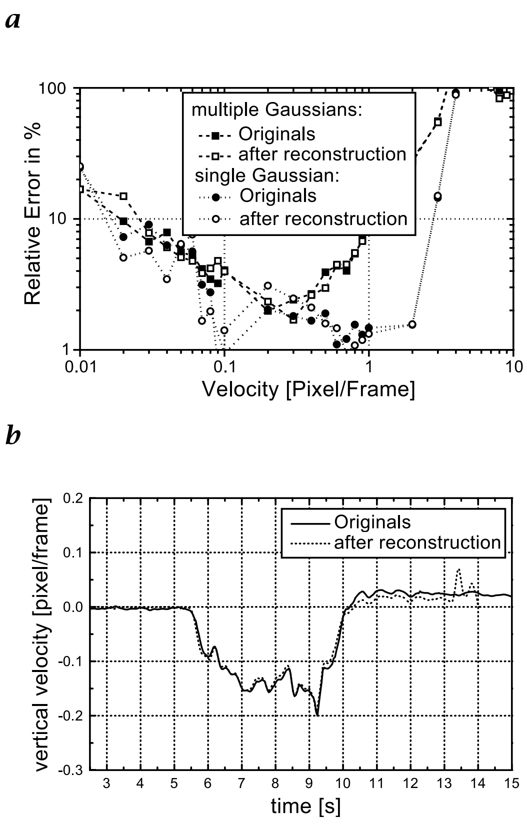


Figure 32.8: *Optical flow estimation with the structure-tensor technique on original images and on reconstructed images after compression:* ***a*** *relative error on generated test data; and* ***b*** *calculated velocity on an image sequence of the sediment boundary layer.*

locity field can be interpolated from the sparse velocity data by means of the *normalized convolution* [Volume 2, Chapter 7]. The coherence measure serves as our confidence measure, where we consider only the previously segmented areas. For effective computation we use the binomial smoothing operator in the form of a Gaussian pyramid for averaging. Empirical tests showed that the second pyramid level gives best results with respect to the trade-off between fill factor and over-smoothing. Figure 32.9a shows the calculated vertical velocity where the coherence measure exceeds 0.5. Figure 32.9b is the result after the convolution. There are still areas where no reliable velocity information can be obtained because here the gaps are too wide to be filled.

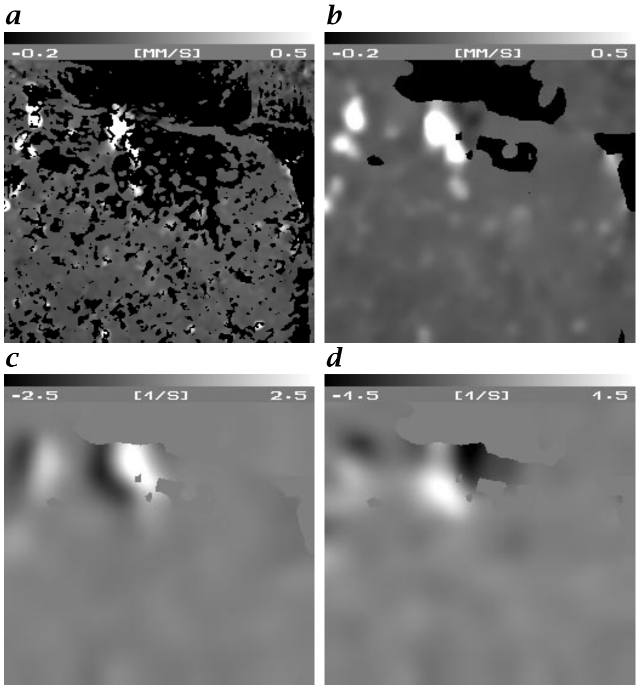


Figure 32.9: *Interpolation of a dense displacement vector field from a sparse field by normalized convolution and calculated characteristics of the velocity field:* ***a*** *sparse vertical velocity;* ***b*** *interpolated velocity;* ***c*** *value of the rotation; and* ***d*** *divergence; (see also Plate 6).*

32.4.5 Mixing parameters

To determine the amount of mixing that occurs during a hydraulic load cycle, two different methods are utilized. One approach relies on the calculation of velocity field parameters such as *divergence* and *rotation*. Second, a special experiment is carried out using color-coded sand particles, which will be described later.

The rotation of a vector field is given by:

$$\operatorname{curl}(\boldsymbol{u}) = \nabla \times \boldsymbol{u} \tag{32.2}$$

The resulting vector stands perpendicular to the image plane, its value characterizes the amount of circular movement, that is, mixing. From the sign follows whether the movement occurs right- or left circular. Figure 32.9c shows an example of the calculated rotation value revealing a zone of strong circular motion directly at the soil-gravel boundary. The relative expansion of soil regions can also be parameterized by the divergence of the displacement vector field

$$\mathrm{div}(\boldsymbol{u}) = \nabla \boldsymbol{u} \tag{32.3}$$

Positive values indicate locally higher velocities than that of the surrounding material, negative values stand for comparatively lower velocities. Both cases lead to a change in the local soil structure.

Mixing from color-coded particles. In order to study structural changes just below the gravel-sand boundary, two areas of color-coded sand were placed beneath the gravel. By using an endoscope, a color camera images the border between a red and a green layer. At the beginning of a measuring campaign, the two areas are divided by a clear-cut line. This distinction gradually ceases to exist until both are nearly completely mixed due to repetitive load changes. For a quantification of the mixing process it is first necessary to distinguish soil particles according to color.

We only use the red and green channels of an RGB camera because only red and green sand has been used. The gains of the two channels are individually adjusted so that the two differently colored sand types show the same brightness.

In RG space, ideal colors are found on lines through the origin. Colored particles, on the other hand, are found to approximately follow a 2-D Gaussian distribution because of statistical variations in their physical properties and orientation. From images where only one type of color is present, calculation of the normalized histogram $h(r,g)$ yields the corresponding probability distributions. The normalized histograms for red, green and uncolored gray sediment are shown in Fig. 32.10b. It is then possible to calculate the inertia tensor:

$$\theta_{r,g} = \sum_{r,g} h(r,g)(\boldsymbol{x}^2 - \delta_{rg} rg) \tag{32.4}$$

Eigenvector analysis of this symmetric tensor yields the two principal axes $\boldsymbol{h}_0$ and $\boldsymbol{h}_1$. We define a new coordinate system were the origin is at the intersection of the two principal axes $\boldsymbol{h}_0^r$ and $\boldsymbol{h}_0^g$ to the lower eigenvalues. The eigenvector of the green distribution serves as one coordinate axis in this coordinate system as shown Fig. 32.10a. In this new coordinate system, we define a dividing line under an angle γ that is used for *segmentation*. This line is chosen so that there is equal fractional probability for the two distributions on opposing sides. This choice minimizes the amount of wrongly classified particles.

In the foregoing, we assume no gray sand to be present in the area under consideration, which can be validated by visual inspection. Problems occur if there are specular reflexes that show no color and come to lie close to the diagonal in RB space. Such instances will be classified

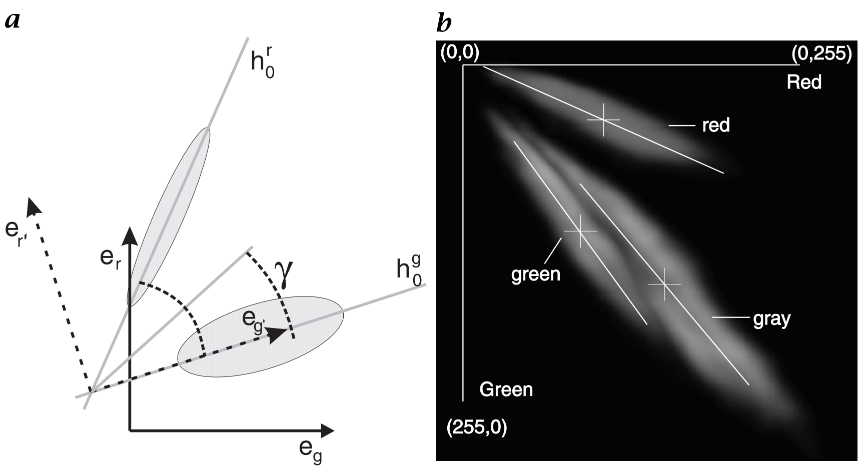


Figure 32.10: *Segmentation on color images:* ***a*** *coordinate transformation in RG space; and* ***b*** *example histograms with principal axes for red, green and gray (not colored) sand.*

as green because the green distribution lies rather close to the diagonal (Fig. 32.10b).

If we describe the mixing of the layers as a *diffusion* process, the particle concentrations $c_i(z,t)$, $i \in \{r,g\}$ along the vertical z-axes follow the diffusion equation

$$\frac{\partial c_i}{\partial t} = D\frac{\partial^2 c_i}{\partial z^2} \tag{32.5}$$

where D is a diffusion constant. Initially, we assume an ideal borderline to be located at position z_0. Thus, we take $c(z < z_0, t = 0) = \text{const}$ and $c(z > z_0, t = 0) = 0$ as boundary conditions. Then, Eq. (32.5) is solved by the following function [6]:

$$c(z,t) = \frac{1}{2}\left(1 - \operatorname{erf}\left(\frac{z - z_0}{\sqrt{2}\,\|\sigma\|}\right)\right), \text{ where } \sigma^2 = 2Dt \tag{32.6}$$

Here, erf denotes the error function, that is, the integral over a Gaussian distribution. Fitting the measured vertical concentration of red/green particles at a given time to Eq. (32.6) using a χ^2 fit, yields the two parameters z_0 and σ. From the first, it is possible to deduce the movement of the complete sediment. Whereas the second does provide information about the width of the distribution, that is, how far the layers have mixed.

At this point, it has to be mentioned that the described method is not very robust. This is due mainly to the fact that there are not enough particles (a sand grain is roughly 15 pixels in diameter) in the viewing

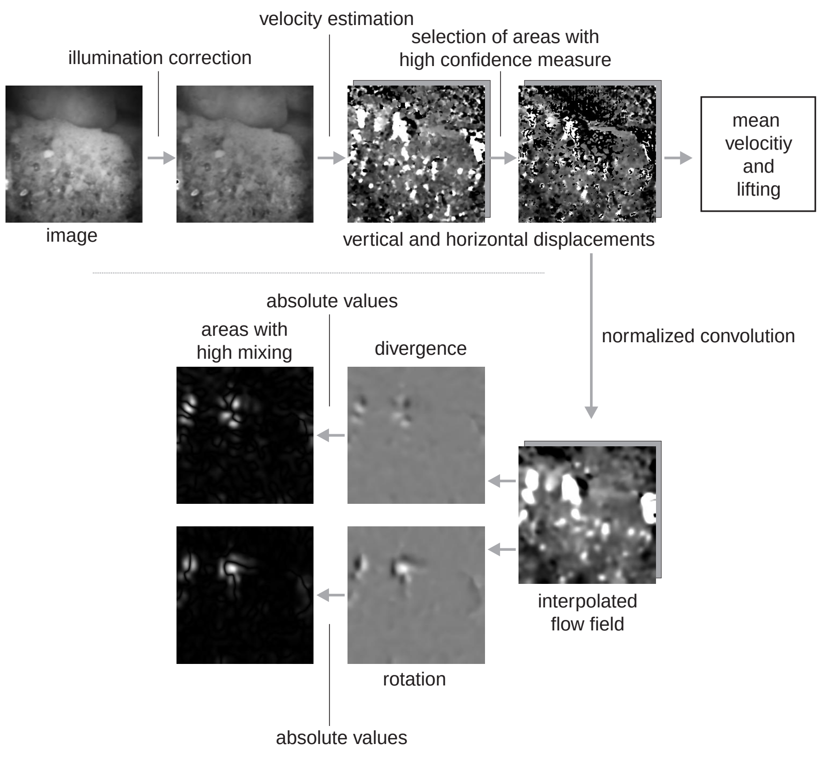


Figure 32.11: *Image processing summary.*

area to provide sufficient statistics. However, this can be moderated in future experiments by enlarging the observed window.

32.4.6 Estimation of water flow

For the estimation of the water velocity field through the gravel cavities, we use the established *particle-imaging velocimetry* (PIV) algorithm [7]. Displacements between two frames are found by searching for the maximum value of their cross correlation [Volume 2, Section 13.5].

In order to study the trajectories of single particles, we use a method known as *particle-tracking velocimetry* (PTV) [8], see Chapter 31. With this approach, particles in one frame are located in the next frame and tracked over a series of images. However, due to the muddy water encountered in the tank it is not possible to segment many particles reliably. Thus, we obtain only sparse velocity information with this method.

32.4.7 Image processing summary

Apart from the flow analysis presented in the previous section, the image processing follows the same steps. The image sequences are compressed, using the algorithm described in Section 32.4.3, just after they were taken and then stored on CD's. After decompression, the images are first corrected for inhomogeneous illumination. Then, the velocity fields are calculated using the structure-tensor algorithm. From these fields, it is already possible to obtain averaged velocities using only those areas where the coherence measure exceeds 0.5. However, for further calculations we need smooth and continuous fields. Those are obtained via the normalized convolution. As a last step, rotation and divergence values are calculated to obtain mixing parameters. Figure 32.11 summarizes the image-processing steps involved.

For the color images location z_0 and width σ of the boundary are calculated additionally as described in Section 32.4.5.

32.5 Results

Some exemplary results are shown in this section together with a simple model that allows an estimation of expected velocities at given pressure situations. (For detailed results see [9, 10]).

32.5.1 Movements due to pressure changes

We typically encounter a water draw down of 60 cm within 5 s after a passing ship in an inland channel corresponding to a pressure change of 60 mbar/s. Thus, we sought to simulate this conditions under different situations. The most important parameter is the mechanical load placed upon the soil structure, which can be simulated by additional hydraulic pressure from above. We studied three possibilities:

- no additional mechanical load: Only the weight of the steel bars and metal sheets, used to distribute the force, is present. This is far too little to prevent the soil from floating due to the entranced air content;
- critical mechanical load: Enough pressure is used to stabilize the soil under normal conditions, that is, pressure corresponding to 4-m water level and a draw down as described in the foregoing. However, already at lower water levels of 2 m, destabilization sets in; and
- very high mechanical load: Under such a holding force the soil should remain stable even for higher draw down and low water levels.

As mentioned earlier, it is the air content in the soil that causes *excess pore water pressure* to build up. This pressure can in turn cause the

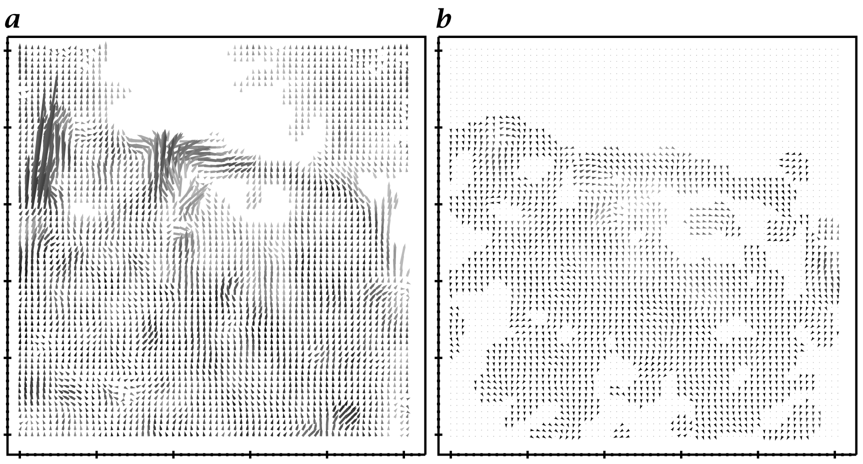


Figure 32.12: *Displacement vector fields:* ***a*** *just after the draw down; and* ***b*** *20 s later.*

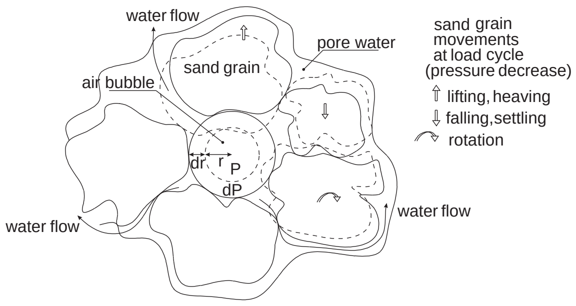


Figure 32.13: *Possible grain movements due to changing hydraulic load conditions.*

sediment to flow like a fluid if there is insufficient mechanical load to stabilize.

The pressure gradient can cause the complete sediment to lift as can be seen from the displacement vector fields calculated with the structure tensor shown in Fig. 32.12. In this example, no additional mechanical load was placed on the sediment. The second velocity field corresponds to a later time where the movement is in the opposite direction as the excess pressure diminishes and the sediment settles down.

32.5.2 Observed and predicted motion

The described lifting will always occur to a certain extent as the air bubbles contained in the sediment expand (see Fig. 32.13). If this ex-

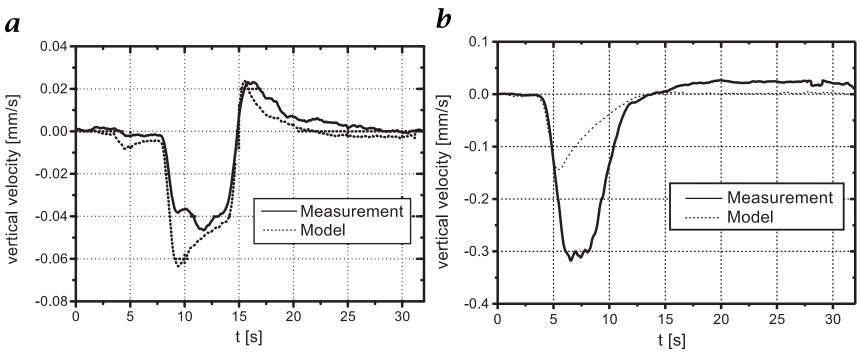


*Figure 32.14: Measured and calculated velocities: **a** stable setup; and **b** unstable situation. Note the different scales.*

pansion is the only cause of motion we can derive a simple model to quantify the accompanying movements. If, on the other hand, the water flow due to expanding air bubbles causes sediment particles to be drawn away, this simple description will cease to appropriately explain the encountered motion.

For the relation between pressure P within the bubble (which equals the pressure of the surrounding liquid) and its volume V, we take the ideal gas equation

$$PV = \nu R T = \text{const} = c \tag{32.7}$$

For constant temperature T, a Taylor expansion of the volume for an infinitesimal pressure change dP, and resubstitution of the bubble radius, the following formula for the expected velocity can be derived [10]:

$$u(t) = \frac{\mathrm{d}r(t)}{\mathrm{d}t} = -(\frac{c}{36\pi})^{\frac{1}{3}}\, P^{-\frac{4}{3}}\, \frac{\mathrm{d}P(t)}{\mathrm{d}t} \tag{32.8}$$

If the expansion of the air bubbles is the only cause of motion, Eq. (32.8) should qualitatively describe the measured velocities. The free parameter c stems from the unknown air content and should remain constant with time.

In Fig. 32.14, the measured mean velocity and the expected velocity are plotted together. In the stable environment, the model agrees surprisingly well with the measured data. In the other case (Fig. 32.14b) the layers do not remain stable and the model no longer captures the situation. However, the onset of the motion is still described correctly. This strongly suggests that the expansion of air in the sediment is the actual cause of motion.

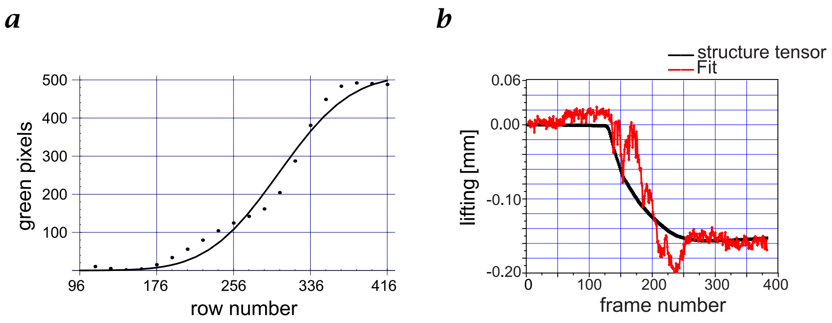


*Figure 32.15: Fitting the color distributions: **a** example fit; and **b** comparison of computed lifting for the tensor method and the fit.*

32.5.3 Mixing

As described in Section 32.4.5, we use divergence and rotation of the velocity field in order to quantify the amount of mixing. It turns out that both values are strongly related. This is not altogether surprising due to the inhomogeneous structure of the sediment. Wherever there are local velocity gradients they will lead to some circular movement. We obtain the expected results that the mean values are about one order of magnitude higher for experiments with lower mechanical load and lower water levels.

Color: An example fit of the measured color distribution to Eq. (32.6) is shown in Fig. 32.15a. The first parameter extracted is the position of the red-green boundary z_0. This also describes the lifting of the complete sediment and can be compared with the result obtained from the integration of the velocities calculated with the structure tensor. Figure 32.15b compares these two approaches; they qualitatively yield the same values.

However, the forementioned problem, due to relatively bad statistics, results in scattered and not very accurate results. Thus, the computed boundary width σ only allows qualitative statements. One observation is that the width does increase by a factor of 2 over the whole measurement campaign. Second, it is observed that within one experiment (one water level draw down) σ remains the same for stable situations and shows some broadening for unstable situations. Thus, the broadening described by $\sigma(t)$ is a function of the numbers of draw downs, rather than a function of t within one pressure change.

32.5.4 Flow in gravel

One example of the flow field within a pore volume is illustrated in Fig. 32.16a. This was calculated at a specific time step using PIV. From

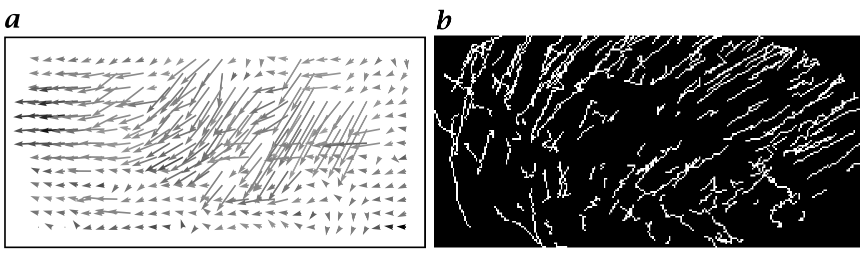


***Figure 32.16:** Water flow in gravel: **a** Eulerian motion vector field at one point in time; and **b** Lagrangian trajectories for a sequence.*

Fig. 32.16b the trajectories of some particles, as computed with the PTV algorithm, over a time of 10 s can be seen.

32.5.5 Results summary

For high loads it is possible to account for sediment movements through the expansion of air bubbles within the soil. Here, a simple model yields qualitatively good results. For lower loads, a fundamentally different situation appears. Velocities are higher by one order of magnitude (see also Fig. 32.14), and the model does not correspond to the measured displacements. The calculated mixing parameters show a similar picture with much higher values for low load conditions. Image sequences taken under such circumstances, clearly show a flow of sediment. This flow of material also becomes evident in the overall lifting of the sediment that is far greater (> 2 mm) for such unstable situations than it is for stable configurations ($0.01 \ldots 0.1$ mm).

32.6 Conclusions and future activities

A novel approach to investigate movements that occur at the boundary of gravel and sediment layers in river sediments was presented. To handle large data sets, we introduced an image-sequence compression scheme based on motion detection. It was shown that this compression does not alter the calculation of optical flow fields computed with the structure tensor using the compressed image sequences. A fast method for color segmentation was presented. From the extracted distributions, conclusions about lifting and mixing within sand layers can be derived.

A combined analysis of the captured velocities and the simultaneously measured pressure values reveals excess pore water pressure to be responsible for soil movements.

We also showed an approach to study water flow within a pore volume. The feasibility of this method could be demonstrated. In the near future, systematic studies of horizontal flow fields in and above gravel layers are planned.

Acknowledgments

We would like to thank Mr. H.-J. Köhler from the Federal Waterways Engineering and Research Institute (BAW), Karlsruhe, for the support, both practical and financial, that made this project feasible.

32.7 References

[1] Köhler, H. J., (1993). The influence of hydraulic head and hydraulic gradient on the filtration process. In *Filters in Geotechnical and Hydraulic Engineering*, Brauns, Heibaum, and Schuler, eds., Balkema. Rotterdam.

[2] Gröning, H., (1996). *Mehrgitterbewegungssegmentierung und Messungen am großen Simulationstank zur Bestimmung von Sedimentverlagerungen.* Diploma thesis, University of Heidelberg.

[3] Haußecker, H., (1993). *Mehrgitter-Bewegungssegmentierung in Bildfolgen mit Anwendung zur Detektion von Sedimentverlagerungen.* Diploma thesis, University of Heidelberg.

[4] Köhler, H. J., Haußecker, H., and Jähne, B., (1996). Detection of particle movements at soil interfaces due to changing hydraulic load conditions, localised by a digital image processing technique. In *2nd International Conference on Geofilters*, J. Lafleur and A. L. Rollin, eds., pp. 215–226. Montreal.

[5] Haußecker, H., Jähne, B., and Köhler, H. J., (1995). Effektive Filterverfahren auf Mehrgitter-Datenstrukturen. In *4. Symposium Bildverarbeitung '95*, Technische Akademie Esslingen. Ostfildern.

[6] Crank, J., (1975). *The Mathematics of Diffusion.* New York: Oxford University Press.

[7] Willert, C. E. and Gharib, M., (1991). Digital particle image velocimetry. *Experiments in Fluids*, **10**:181–193.

[8] Hering, F., Merle, M., Wierzimok, D., and Jähne, B., (1995). A robust technique for tracking particles over long image sequences. In *ISPRS Intercommision Workshop "From Pixels to Sequences"*, E. P. Baltsavias, ed., pp. 202–207. Coventry UK: RICS Books.

[9] Beringer, O., (1998). *Ein Meßsystem zur Untersuchung von Sedimentverlagerungen und Durchmischungsprozessen mittels digitaler Bildfolgenanalyse.* Diploma thesis, University of Heidelberg.

[10] Spies, H., (1998). *Bewegungsdetektion und Geschwindigkeitsanalyse zur Untersuchung von Sedimentverlagerungen und Porenströmungen.* Diploma thesis, University of Heidelberg.

33 Plant-Leaf Growth Studied by Image Sequence Analysis

Dominik Schmundt[1,2], and Ulrich Schurr[2]

[1]Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR)
Universität Heidelberg, Germany
[2]Botanisches Institut, Universität Heidelberg, Germany

33.1 Introduction

We describe here the application of basic algorithms of image-sequence analysis to detect the growth of plant leaves. The aim of this work is to provide a tool to measure areal growth of a *plant leaf* at high spatial and temporal resolution.

Our technique uses a low-level motion analysis to obtain displacement vector fields for each step of the sequence [1]. Maps of the growth

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

rate can be derived mathematically from the displacement vector field by calculating its divergence as the sum of its spatial derivatives in the x- and y- directions.

Leaf growth is a fundamental process for the performance of the plant. Distribution of growth over the plant determines its habitus and the structure of the tissues themselves are laid down during the growth process and can hardly be altered thereafter. In parallel to growth, leaves differentiate and successively gain the function of mature tissues. Analysis of growth, therefore: (i) is an essential prerequisite for studying basic growth processes; (ii) provides the framework for analysis of cytological, physiological, biochemical and molecular functions; and (iii) is important in understanding the adaptation of the plant to its environment. However, because growth is not uniformly distributed over the leaf blade in time and space, the analysis requires methods with spatial and temporal resolution simultaneously, a property that is inherent in the approach to image-sequence analysis.

33.2 Previous investigations

Quantitative investigations of *plant growth* can roughly be categorized as in the following scheme (the items are explained in more detail in the following)[2]:

- direct measurement by hand, based on *biometric relations*;
- destructive methods for biomass analysis as, for example, fresh and dry weight of the plant;
- continuous measurements using mechanical length transducers; and
- optical techniques using photographs of the plant that are evaluated by hand (sometimes using image processing to compare consecutive images).

In areally growing tissues, quantitative analysis is often based on biometric relationships for the lack of other more appropriate methods, for example, the ratio between the product of leaf width and leaf length to the leaf area. This ratio is often constant within one species and can easily be determined. However, sufficient temporal resolution is hardly obtained in such studies. Direct estimation of area can be done with commercially available leaf-area meters, but this often involves detachment of the leaf and thus destructive sampling. Alternatively, simple threshold methods can be used for segmentation of the leaf area on video or digitized images. Area is then calculated from a calibration of pixel per unit area. These techniques usually require the detachment of the leaf from the plant. However, in more sophisticated setups, growth can be analyzed in attached dicot leaves by video imaging and determination of the leaf area by image analysis. The accuracy of such systems is sufficiently high to determine variations of the increase in the leaf area within minutes, as, for example, in response to air humidity [3].

None of these integrating techniques is able to analyze the nature of the changes in growth rates. Because growth rate at the tissue is the integral of the expansion of individual cells within the growing zone, changes in the growth rate can be caused by variation in cell elongation rates as well as by the size of the expanding area. It has been shown in growing roots that environmental factors can influence expansion rate via either of these mechanisms [4]. On the other hand, distribution of growth rates can change quite dynamically within minutes [5], and this has functional relevance, for example, in directing growth in response to environmental stimuli such as gravity in roots (gravitropism [6]). It is, therefore, obvious that a thorough analysis of growing tissue needs to be based on spatial and temporal information about the growing zones.

In linearly arranged growing zones, spatial information on expansion rates can be obtained by tracking series of landmarks along the growth zone. The quality of the landmarks used in such studies ranges from punctures with fine needles [7, 8] to inert carbon particles [6] depending on the accessibility of the growing zone. The expansion rates of the tissue between these landmarks can then either be analyzed by determination of the change of distance between the marks after a certain time or, if the growing zone is visually accessible, by continuous observation, for example, by video imaging.

In areally organized growth zones it is much more difficult to obtain spatial information. Landmark techniques can be employed in an analogous manner as described in the foregoing. However, many more measurements are needed to obtain the same resolution. In addition to the increased effort involved in obtaining the basic data set, growth in an areally growing region is more difficult to analyze for formal reasons. In a linear system, all landmarks move in the same direction, that is, all growth vectors have the same angular component and, usually, it is easy to determine a distinct point of reference (roots: root tip, leaf: leaf tip) where the integrated elongation rate can be measured. However, in areally growing parts, the data set obtained from landmark distortion is a vector field in which the individual growth vectors may have quite different values and directions. A good example of such a very tedious work is given by Wolf et al. [9], who determined maps of growth intensities and directions of pathologically malformed vine leaves. One possibility of circumventing this problem is to determine areal growth rates directly [10]. This can be done by planimetry of the interveinal areas in time-lapse image sequences of leaves [11]. Patterns of artificial landmarks can also be used to study distribution of growth rates [12], however, these systems have the significant drawback that the thus introduced coordinate system has nothing to do with the physiological nature of the leaf. Recent studies investigated *seedling growth* by image-sequence analysis [13, 14]. This chapter presents a technique for analysis of dicot *leaf growth* with high spatial and temporal resolution.

33.3 Experimental setup

The development of the analysis was done in close interaction between the development of the setup best suited to acquire appropriate image sequences and the programming of the image analysis software including mechanical stability (Section 33.3.1) and illumination of the scene (Section 33.3.2). The software was written in a macro language for image-sequence analysis (heurisko) and was eventually spilt into four different modules:

- the acquisition module acquires image sequences (Section 33.3.3);
- the calibration module (Section 33.4.1) does the geometric calibrations that are passed over to the visualization module and interactive threshold adjustments prior to the actual motion estimation in the evaluation module (Section 33.4.1). These parameters are set in this module so that the evaluation can easily be repeated with the same set of parameters on a different time base;
- the evaluation module calculates (Section 33.4.2) and interpolates (Section 33.4.3) the displacement vector field (DVF) of the image sequence in a batch mode based on the input from the calibration module; and
- the visualization module (Section 33.4.4) that provides procedures to interactively browse through the raw and calculated data, extracts velocity and growth profiles.

33.3.1 Mechanical setup

For the determination of motion it is necessary that the motion imaged in the image sequences originates only from the object and not from the setup itself. Thus, to suppress displacement of the camera relative to the leaf the entire setup was constructed from heavy stands (Fig. 33.1). A classical continuous length measurement was included in the setup for evaluation of the method. Additionally, this fixed the leaf base and the main axis of growth was clamped to the focal plane by means of a nylon twine attached to the leaf tip by a small clamp holder. The twine was kept under tension by a weight bound to its end and was led over a reel connected to a *rotary voltage transducer*, which continuously monitored the movement of the reel. This allowed the monitoring of minute displacements. With a rotary resolution of 11 bit (2048 steps) and a reel of 15 mm diameter, the minimal displacements to resolve were of the magnitude of 25 μm.

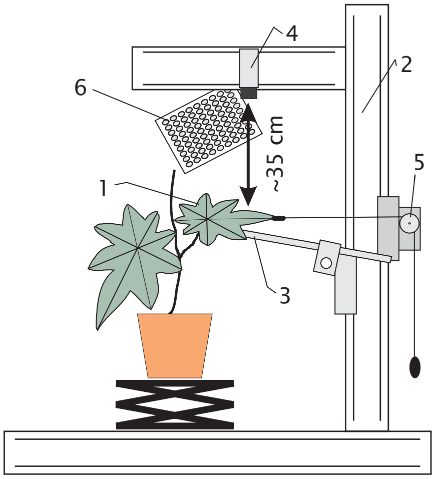


Figure 33.1: *The mechanical setup for the optical measurement of leaf growth: (1) a* Ricinus *plant on an adjustable stage; (2) frame of heavy stands; (3) small rod for the fixation of the leaf base; (4) CCD camera; (5) rotary voltage transducer for length measurement; (6) panel of IR LED for illumination.*

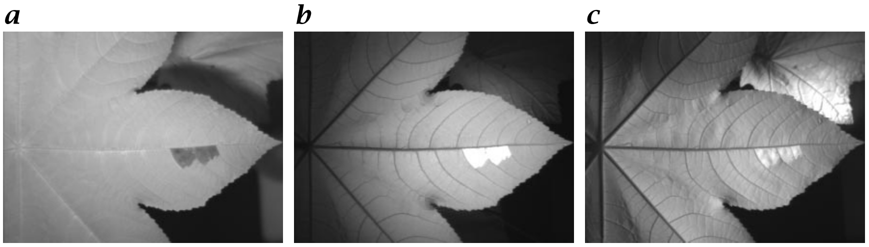


Figure 33.2: *A Ricinus leaf illuminated by LEDs at 950 nm in the three different modes:* ***a*** *reflection;* ***b*** *transmission; and* ***c*** *lateral scatter. The spot on all images is the result of an infiltration of the sponge parenchyma with water, closing the intercellular air spaces responsible for the scattering.*

33.3.2 Illumination

For the illumination of the leaf, different light sources and illumination modes have been carefully considered. For motion analysis, a signal is desired that is rich in contrast, closely correlated to the leaf structure and insensitive to small deviations of the orientation of the leaf surface. An additional constraint is introduced by our application: the illumination must not cause any response of the leaf. The latter condition narrows the choice for a light source to the wavelength beyond 900 nm because apart from the absorption of *chlorophyll*, which peaks at 430 nm and 680 nm, the leaf contains pigments absorbing light of wavelength up to 880 nm that are involved in morphogenetic[1] pathways.

In this wavelength range, GaAs LEDs with a narrow emission peak around 950 nm provide a stable light source, which matches well with the sensitivity range of the standard silicon CCD (500 nm to 1100 nm) (see also Volume 1, Chapter 7). To suppress the contribution of the *growth chamber* illumination to the signal and to avoid specular reflection that occurs in the visible light, a long-wavelength pass filter was placed in front of the camera lens.

To understand the nature of the leaf image in near IR illumination, it is necessary to look at the leaf structure and its optical properties. Roughly, the leaf is built of four layers of different cells: (i) the upper epidermis[2]; (ii) the palisade parenchyma[3]; (iii) the spongy parenchyma; and (iv) the lower epidermis. Additionally, the leaf is covered by the cuticle, a waxy layer on the upper and lower epidermis. Contrary to the other layers, the sponge parenchyma is loosely packed containing many small air-filled cavities. The near IR is transmitted without absorption through the dense cell layers and then scattered by the sponge parenchyma (see Vogelman et al. [15] for a thorough review of the optical properties of leaves). The diffuse scattering by the sponge parenchyma results in a signal that is very close to the reflection from a Lambertian surface, considered ideal for image-processing applications, which is proved by the change in the optical properties when the leaf is infiltrated with water (Fig. 33.2).

Three different modes of illumination are possible (Fig. 33.2): **a** backscatter; **b** transmission; and **c** lateral scattering. For compactness of the setup, the backscatter mode was chosen where the leaf is illuminated by a panel of 100 LEDs from the viewing direction.

33.3.3 Image and data acquisition

The images were acquired using a standard CCD camera (Sony XC-75) connected to a gray-scale PCI bus framegrabber (Eltec PC-Eye 1) in a PC (Pentium 133 MHz). The software used for the image acquisition was a standard image-processing package (heurisko) on a Windows 95 platform. The data from the length detector were collected by a second PC with an A/D converter card at a sampling interval of 20 s using a standard data acquisition software (Turbo Lab). For the evaluation of the technique and the determination of the optimum sampling interval, the images were sampled at a rate of one frame/min for diurnal analysis of growth (Section 33.6.1). A full diurnal sequence extends over 24 h and consists of 1440 images. During root-pressurizing experiments

[1]Morphogenetic refers to the genetic program to alter the morphology of a plant.
[2]Epidermis is the name for the outer cell layer of a tissue.
[3]Parenchyma is the name for the inner cell layers of a tissue.

(Section 33.6.2) a sampling interval of 10 s was chosen due to the rapid changes expected and the shorter duration of the experiments (1.5 to 2 h, 540 to 720 images).

Though the base of the leaf is tightly fixed in the setup, shaking of the lobe occurs due to the wind necessary for the climatization of the growth chamber. The wind induced a shaking of a frequency of approximately 1 Hz. As the wind can not be turned off during the acquisition of images, a strategy had to be found to circumvent the violation of the sampling theorem when images are taken at a rate of 1 min^{-1}. By integrating over a sequence of 80 images of a sequence acquired at standard video rate (30 frames/s), the effect of the shaking could largely be eliminated. Additionally, the integration led to an enhancement of the signal-to-noise-ratio of the images (integration over 80 images means an improvement of $\sqrt{80}$), making evaluation at lower light levels feasible. Sequences could thus be acquired with the same aperture during night and day without a need to adjust the brightness of the illumination.

33.4 Image analysis

Measuring growth quantitatively at a high temporal resolution is a difficult task because the displacements that need to be detected are negligible. A young leaf of *Ricinus* of 80 mm grows at a maximum elongation rate of 2 mm/h. To obtain a temporal resolution of 5 min, the system must have a spatial resolution of at least 2 mm/12 = 0.167 mm to detect the movement of the leaf tip. If the entire leaf is imaged on a charge coupled device (CCD) of 640×480 pixels at a sampling rate of 1 frame per 2 min (larger sampling intervals will eventually lead to an incoherent motion, see Section 33.5), the maximum movement of the leaf corresponds to a displacement of 0.53 pixel/frame. Therefore, a motion estimator of an accuracy well in the subpixel range is needed.

33.4.1 Calibration

The calibration module consists of two principal parts: the geometric calibration itself and a part that provides some basic settings for the subsequent evaluation. Additionally, the normalization of the mean gray value as a preprocessing step for the low-level motion estimation is described in this section.

Geometric calibration. For accurate determination of the velocities and the growth of the leaf, the magnification of the system (camera and frame grabber) must be known. Because the leaf is constrained to a plane perpendicular to the direction of view, only the magnification and the aspect ratio and not the full set of 3-D coordinates of the

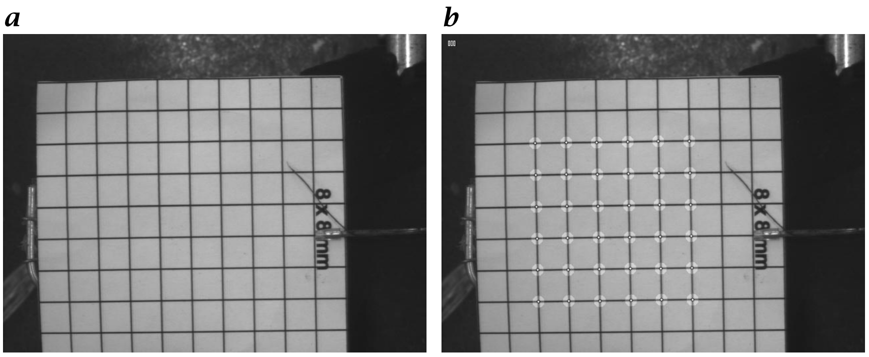


Figure 33.3: ***a*** *The grid used for calibration; and* ***b*** *detected points marked on the grid and camera parameters.*

camera have to be determined. As the growth analysis shall be applied at different scales, an easy-to-handle procedure was chosen: A regular grid printed on a cardboard is placed in the fixation replacing the leaf and a set of calibration images is taken (Fig. 33.3a). For the calibration, the origin, the x- and y- base vectors, the number of grid points ($\mathrm{n}_x \times \mathrm{n}_y$), and the spacing of the grid must be given. Then all grid points are extracted automatically to subpixel accuracy by fitting a cross of Gaussian lines around each grid point [16] (Fig. 33.3b). From the list of points, the magnification, aspect ratio and a parameter for the radial distortion are calculated and the results are passed on to the visualization module (Section 33.4.4) that converts the velocities from pixel/frame coming from the evaluation module to millimeter per hour.

Normalization of the mean gray value. Low-level motion estimators are very sensitive to changes of brightness over time. It is, therefore, necessary to provide either a homogeneously illuminated scene or a reference for the compensation of fluctuations of the image intensity. In our setup the illumination by LEDs provides a stable light source as can be seen in the plot of the mean gray value of a night sequence of *Ricinus* (Fig. 33.4a). Because the growth chamber illumination also emits in the near IR, significant contribution of chamber light occurs during the illumination period (Fig. 33.4b). It can be clearly seen that the ambient light introduces strong fluctuations of the mean gray value that severely hamper the motion estimation as can be judged by the confidence fill factor defined in Section 33.4.2 (Fig. 33.4c). To overcome this disturbance, we utilize the fact that: (i) the motion of the leaf is small, that is, less than 1 pixel/frame and that; (ii) the leaf exhibits a very homogeneous texture. Under these conditions, the variation in the mean gray values of the leaf are due mainly to fluctuations of the illumination and can thus be used to eliminate these. For the correction,

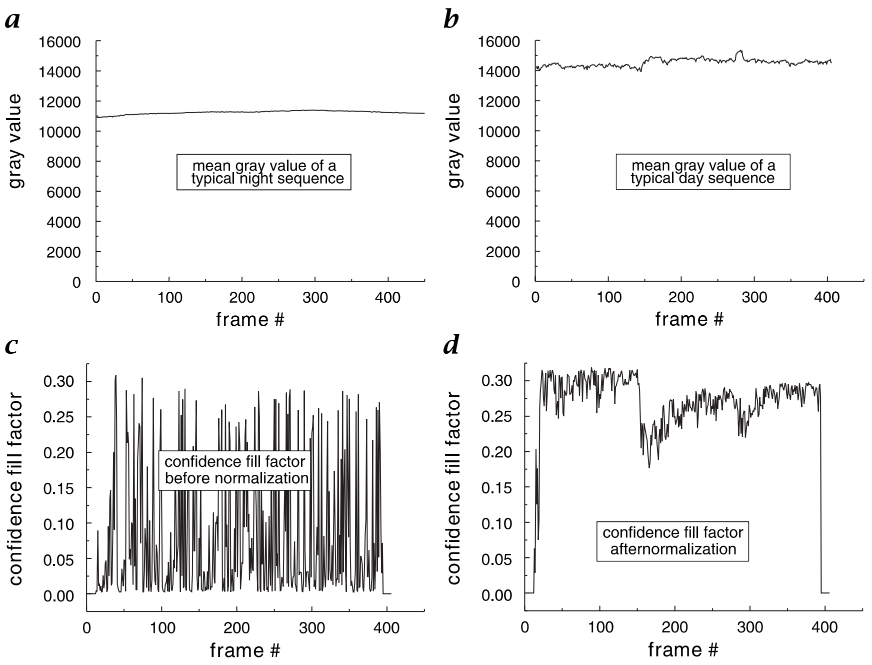


***Figure 33.4:** Plots of the mean gray value of: **a** a night-time sequence (illuminated by an IR LED panel); and **b** a daytime sequence. The variations of image brightness introduced by the growth chamber illumination are obvious. In **c** the confidence fill factor resulting from a direct evaluation of the sequence plotted in **b** is shown (see Section 33.4.2 for the definition of the confidence fill factor). Plot **d** shows the confidence fill factor of the same sequence when the mean gray value is normalized.*

an area of interest is chosen in the center of the image. In this area, the mean gray value (g_{mean}) is calculated and then the image is multiplied by the g_0/g_{mean} resulting in a much higher percentage of valid information from the motion estimator (Fig. 33.4d).

33.4.2 Motion estimation

Motion is estimated by the so-called structure tensor approach [1]. To obtain a dense displacement vector field, local motion is determined by an analysis of the orientation of the gray-value structure of a small spatiotemporal neighborhood ($7 \times 7 \times 7$ pixel for the data presented here) of the image sequence. In the first step, the structure tensor (see Volume 2, Section 13.3.2) is calculated using a Sobel-type filter that is optimized for isotropy of the derivation [17] and a standard binomial smoothing filter for the summation over the spatiotemporal neighborhood. At this point, the low-pass property of the smoothing is utilized

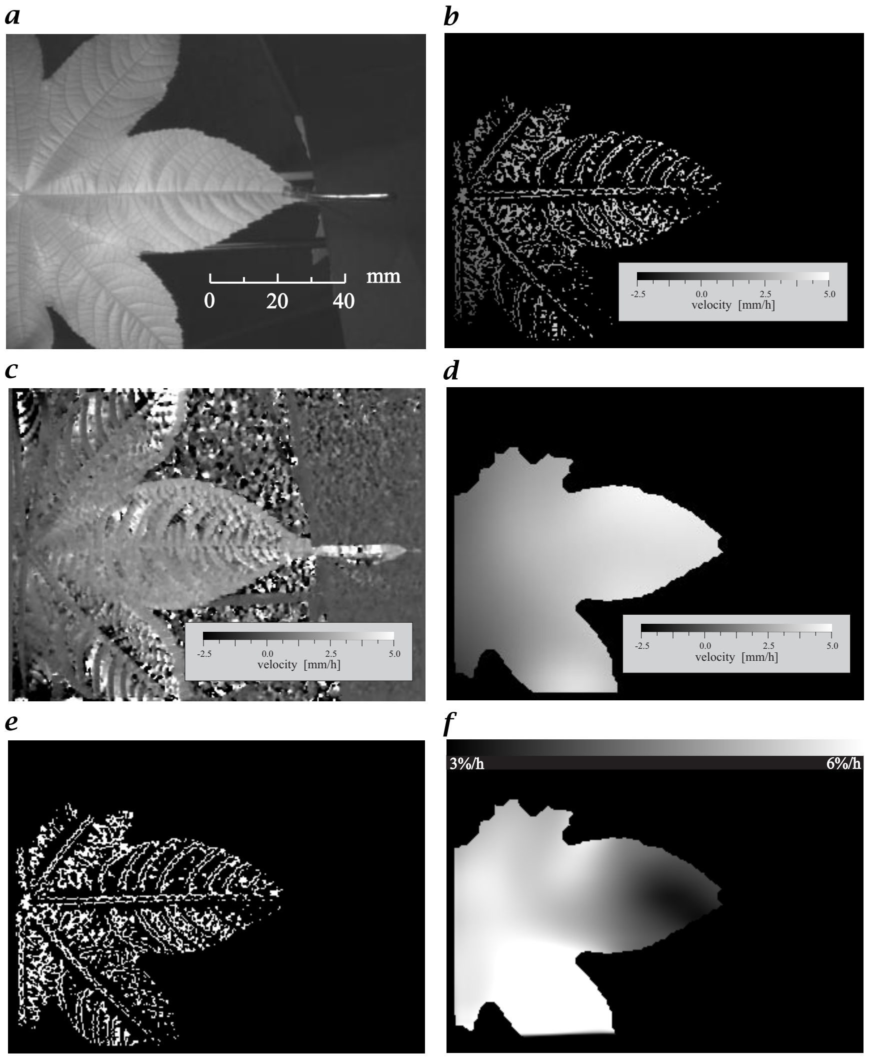


Figure 33.5: *Six images from the course of a full 2-D growth evaluation:* ***a*** *image from the original time-lapse sequence;* ***b*** *raw velocity field (x-component, color coded) of the leaf surface as calculated from the sequence by a low-level motion estimator (see text);* ***c*** *subset of pixel with a confidence measure above a set threshold;* ***d*** *velocity field when* ***c*** *is considered (masked by the leaf mask);* ***e*** *interpolated velocity field (x-component);* ***f*** *divergence of the velocity field (x- and y-direction); (see also Plate 7).*

in the spatial direction to subsample the images, thus obtaining a reduction of the data by a factor of 4, which speeds up all subsequent operations by the same factor.

Analysis of the structure tensor as described in Section 13.3.2 yields the velocity field and a set of control parameters, namely, *coherence*, *edge*, and *corner measure*. The *coherence* indicates whether the algorithm has computed any motion information at all; *edge* and *corner* report on the presence or absence of the aperture problem, respectively. The aperture problem circumscribes the fact that for a linear feature (e. g., a straight line), only the motion perpendicular to its orientation can be determined from focusing at a small neighborhood [18].

At this point, two masks are generated for further analysis: (i) a mask of the borders of the leaf used for the analysis; and (ii) a mask of the pixels considered valid by the motion analysis. The first mask is obtained by simply masking the pixel above a set threshold (usually between 50 to 60 % of the maximum gray value) of the original image and then selecting the largest connected region. This approach yields good results for more than 90 % of the images of a sequence. The mask of the valid motion estimates is obtained by masking out all pixels with a *corner*-measure below 0.8 (a value yielding reasonably dense motion fields). The ratio of the number of valid pixels from the motion analysis to the number of pixels of the segmented leaf area is then defined as the *confidence fill factor*. This is a good estimate of the quality of the motion analysis of an entire sequence.

33.4.3 Interpolation of sparse displacement vector fields

The computation of the 2-D *displacement vector field* (DVF) by the structure-tensor technique can only be performed accurately on the subset of the image where the corner measure exceeds a set threshold (see Section 33.4.2). The DVF resulting from the motion estimation is sparsely and inhomogeneously filled (Fig. 33.5c). For further analysis, a dense and homogeneously filled velocity field must be interpolated from the sparse velocity data. This can be done by using a modified version of the normalized convolution proposed by Knutson and Westin [19] (see Section 7.6), where the so-called applicability function is replaced by a binomial smoothing mask. The binomial mask offers the advantage of separability and thus computational speed (another efficient implementation can be found in Section 37.4.2). For effective computation the binomial operator is implemented by a Gaussian pyramid. We found that smoothing to the fourth pyramid level gives best results for our image material.

As weighting mask we use an image with all pixels with a corner value above 0.8 set to 1. No further weighting is used because the normalized values for the corner and coherence measures, though useful for selecting the pixel of valid information, are not appropriate for error propagation by direct weighting. Figure 33.5d shows the calculated x-velocity where the coherence measure exceeds 0.8; the image in Fig. 33.5e is the result after the convolution.

33.4.4 Visualization of the results

As an integral part of the system, an interactive tool for the display of the results has been developed. First, the sequence of DVF can be browsed and all intermediate results can be displayed. At any point the part of the sequence on which the current result is based can be displayed, which helps to understand outliers in the analysis.

For examples of the output of the visualization module refer to the figures in this chapter. An xt-image of the velocity or divergence (growth) along an arbitrary horizontal axis can be shown in gray value or pseudo-color mode (see Fig. 33.7 for an example of a color-coded plot). The confidence fill factor can be displayed to assess the quality and success of the evaluation (Fig. 33.6a). The DVF can be displayed in different modes and as an overlay to the original image (Fig. 33.5b–f). The velocity profile of any point on the leaf can be plotted and compared to that of the mechanical length sensor (Fig. 33.6b-d).

33.5 Stability and validation

The stability of the optical growth analysis has been investigated by examining the effect of variations of the sampling interval. When choosing an appropriate sampling rate for the growth sequences, two requirements must carefully be balanced: (i) the motion must be coherent over the entire length of the part of the sequence upon which the motion estimation is based; and (ii) the motion of the leaf should fall in the optimum detection range of the motion estimation. For the purpose of finding the optimum *sampling rate* we have sampled sequences at a rate of 1 frame/min and analyzed the sequences at steps of 1 to 4 min. The results of this evaluation are summarized in Fig. 33.6.

The confidence fill factor defined in Section 33.4.2 is a good estimate for the coherency of the motion over time. For the evaluation of our test sequence it is plotted in Fig. 33.6a for the four different time bases. Together with the motion data of the leaf tip, which is the fastest moving point in the image (Fig. 33.6d), it becomes clear that the coherency decreases with an increasing sampling interval. This effect becomes even more pronounced when the velocities are high.

From Fig. 33.6b and c an estimate of the accuracy of the velocity estimation can be given. From the investigations of Haußecker and Jähne [1], we know that the structure-tensor approach works best for displace-

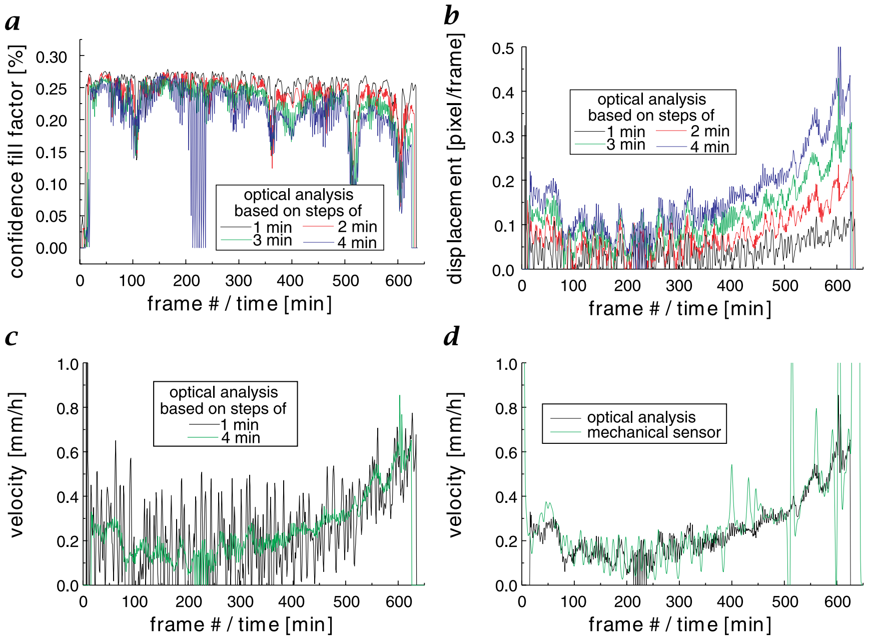


Figure 33.6: *Velocities at different time steps.*

ments between 0.1 to 1.0 pixels/frame. Considering that the velocity at the leaf tip is the maximum velocity in the image, Fig. 33.6b shows that at a sampling interval of 1 min, the maximum velocity becomes too small to be detected properly. In Fig. 33.6c, it becomes clear that the data from the motion estimation becomes noisier with the shorter sampling intervals.

As an independent measurement, the data of the mechanical length detector integrated into the setup of the optical growth analysis was recorded simultaneously during all measurements. This allows us to determine the velocity of the leaf tip mechanically and optically. Comparison of both results confirms the result of the optical motion estimation at the leaf tip (Fig. 33.6d).

Because there are no other techniques that yield growth maps with a high temporal resolution, the growth maps could only be compared qualitatively to the growth distribution known from the literature [10]. Though the data from these sources were sampled on a daily basis, the finding of a distinct region of growth near the leaf base agrees well with our data.

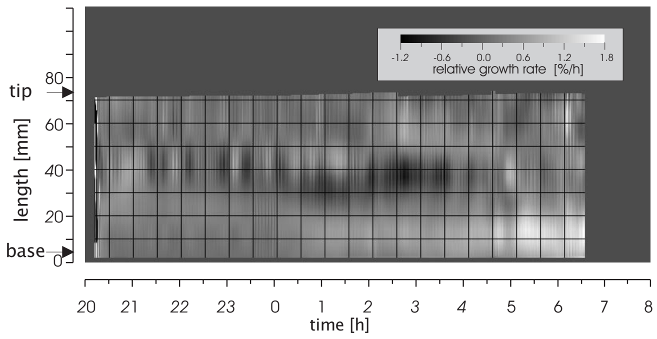


Figure 33.7: *The growth of the middle part of a Ricinus leaf depicted as a spatiotemporal image; (see also Plate 8).*

33.6 Applications

In our preliminary studies, we applied the system to two different issues. First, we investigate the *diurnal variation* in plant growth, and, second, rapid changes in growth pattern induced by step changes of the *turgor pressure*.

33.6.1 Diurnal variations of growth

Diurnal variations of growth and growth pattern are a basic issue when studying the growth of plants. Hence, the first application of the new technique was to monitor the growth of *Ricinus* in a growth chamber. Growth chambers offer the advantage of a well-controlled environment because the lighting, the temperature, and the humidity can be controlled. In growth-chamber experiments, we have investigated the growth of *Ricinus* and *tobacco*. A typical result from a nighttime sequence of *Ricinus* is shown in Fig. 33.7. The temporal and spatial resolutions in this example are approximately 5 min and 5 mm, respectively.

33.6.2 Turgor-induced variations of growth

When looking at the basic process of growth, a clear distinction between cell division and cell elongation must be made. Cell elongation can be described by the commonly accepted *Lockhart* equation [20]

$$\frac{1}{L}\frac{\mathrm{d}L}{\mathrm{d}t} = m(P - Y) \tag{33.1}$$

(This expression denotes the relative elongation rate; m is the extensibility of the cell wall, P is the turgor pressure, and Y is a threshold value for P below which responses of L to P are elastic.) The turgor pressure builds up by the imbalance of osmotic potential inside and outside of the cell and the fact that the cell wall is rigid. Turgor pressure can be of a magnitude as high as 1 MPa. Expansion is initiated by cell wall loosening enzymes that lead to a plastic deformation. To determine the elastic and plastic contribution to deformation a setup by Passioura [21] can be used: The seedling is planted in a “pressure-bomb” so that the root system can be pressurized and the stem of the grown-up plant seals the container. When pressure is applied to the root system the turgor is raised accordingly (0 to 0.4 MPa).

The application of a step changes in pressure at the root system (0 to 20 min: atmospheric pressure, 20 to 40 min: 0.3 MPa above atmospheric pressure, 40 to 60 min: atmospheric pressure) revealed a plastic deformation of the entire leaf blade for a duration of only 5 min. This indicates an active control of the actual growth rate. When the pressure was lowered, a shrinkage of the leaf could be observed that was about one-tenth that of the previous expansion.

33.7 Outlook

Future work on optical growth analysis will include an extensive deployment in the field, for example, a thorough investigation of diurnal rhythms of *Ricinus* and *tobacco*. Also, thorough investigations on the macroscopic extensibility of plant tissue by means of Passioura-type pressure experiments are planned. The possibility of using the same algorithms on other species (for example, *arabidopsis*) will be examined. A comparison with particle tracking approaches in Chapter 31 will be used for further validation of the results and as a complementary tool in situations where the plain velocity estimation is not possible, for example, due to the lack of natural texture in image.

Destructive sampling guided by online mapping of leaf growth will prove to be highly relevant in elucidating the underlying mechanisms on the biochemical and physiological level of growth control. On the road to an integrated system for the remote sensing of physiological data we will combine the optical growth analysis with, for example, the simultaneous observation of *green fluorescent proteins* (GFP), which may be used as a guide for various biochemical activities. Another probe for intracellular Ca^{2+} concentration is described in Knight et al. [22] (for an introduction to the potential of these techniques, see Chapter 34). Furthermore, the IR radiation of the leaf can be used to obtain information on the water status of the leaf (see Chapter 36).

The approach described in this chapter is limited to the analysis of leaves in planar setting. Motion out of the plane that occurs in any natural situation severely biases the results. To solve this problem, a

3-D representation of the leaf will be obtained from approaches such as depth-from-motion or depth-from-stereo (see Chapter 17). With the full 3-D analysis, it will be possible to analyze growth and plant motion at the same time.

Acknowledgments

The authors thank the German Science Foundation (DFG) for financial support of the research initiative "Study of dynamic processes by image sequence analysis." Ulrich Schurr is grateful for initial support of this work by the German Science Foundation (DFG) through SFB 199, project C1.

33.8 References

[1] Haußecker, H. and Jähne, B., (1997). A tensor approach for precise computation of dense displacement vector fields. In *Mustererkennung 1997*, pp. 199–208. DAGM.

[2] Schurr, U., (1997). Growth physiology: Approaches to a spatially and temporally varying problem. *Progress in Botany*, **59**:355–373.

[3] McDonald, A. J. S. and Davies, W. J., (1996). Keeping in touch: responses of whole plant to deficits in water and nutrient supply. *Adv. Bot. Res.*, **22**: 229–300.

[4] Silk, W. K., (1992). Steady form from changing cells. *Int. J. Plant Sci.*, **153**: 49–58.

[5] Frensch, J. and Hsiao, T. C., (1994). Transient responses of cell turgor and growth of maize roots as affected by changes in water potential. *Plant Physiol.*, **104**:247–254.

[6] Buff, E., Baake, M., and Sievers, A., (1987). An empirical function for the description of root growth. *Plant Physiol.*, **83**:685–690.

[7] Dodd, I. C. and Davies, W. J., (1996). The relationship between leaf growth and ABA accumulation in the grass leaf elongation zone. *Plant Cell Environ.*, **19**:1047–1056.

[8] Schnyder, H., Nelson, C. J., and Couts, J. G., (1987). Assessment of spatial distribution of growth in the elongation zone of grass leaf blades. *Plant Physiol.*, **85**:290–293.

[9] Wolf, S. D., Silk, W. K., and Plant, R. E., (1986). Quantitative patterns of leaf expansion: comparison of normal and malformed leaf growth in *Vitis vinifera* cv. Ruby Red. *Am J Bot*, **73**:832–846.

[10] Maksymowych, R., (1973). *Analysis of Leaf Development.* Cambridge University Press, Cambrigde.

[11] Taylor, G., Ranasinghe, S., Bosac, C., Gardner, S. D. L., and Ferris, R., (1994). Elevated CO_2 and plant growth: cellular mechanisms and responses of whole plants. *J. Exp. Bot.*, **45**:1761–1774.

[12] Granier, C. and Tardieu, F., (1998). Spatial and temporal analysis of expansion and cell cycle in sunflower leaves. *Plant Physiol.*, **116**:991–1001.

[13] Barron, J. L. and Liptay, A., (1994). Optic flow to measure minute increments in plant growth. *Bioimaging*, **2**:57–61.

[14] Barron, J. L. and Liptay, A., (1997). Measuring 3-D plant growth using optical flow. *Bioimaging*, **5**:82–86.

[15] Vogelman, T. C., Nishio, J. N., and Smith, W. K., (1996). Leaves and light capture: light propagation and gradients of carbon fixation within leaves. *Trends In Plant Sciences*, **1**(2):65–70.

[16] Schultz, M., (1997). *Geometrische Kalibrierung von CCD-Kameras*. Diploma thesis, Universität Heidelberg.

[17] Scharr, H., Körkel, S., and Jähne, B., (1997). Numerische Isotropieoptimierung von FIR-Filtern mittels Querglättung. In *Mustererkennung 1997*, pp. 367–374. DAGM.

[18] Jähne, B., (1997). *Digital Image Processing*. New York: Springer.

[19] Knutson, H. and Westin, C.-F., (1993). Normalized and Differential Convolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 515–523. New York: IEEE.

[20] Passioura, J. B. and Fry, S. C., (1992). Turgor and cell expansion: beyond the Lockhard equation. *Journal of Plant Physiology*, **19**:565–576.

[21] Passioura, J. B., (1994). The physical chemistry of the primary cell wall: implications for the control of expansion rate. *Jour. Experimental Botany*, **45**:1675–1682.

[22] Knight, M. S., Campbell, A. K., Smith, S. M., and Trewavas, A. J., (1991). Transgenic plant aequorin reports the effect of touch and cold shock and elicitors on cytoplasmatic calcium. *Nature*, **352**:524–526.

34 Mathematical Modeling of Ca^{2+}-Fluorescence Images

Dietmar Uttenweiler and Rainer H. A. Fink

II. Physiologisches Institut, Universität Heidelberg, Germany

34.1 Introduction

The development of very powerful *fluorescence imaging* techniques has led to a much deeper understanding of the structural and functional organization of cellular systems. Especially the possibility of measuring intracellular *ion concentrations* with high spatial and temporal resolution (Chapters 21 and 12) has contributed to our knowledge about the mechanism of cellular information transduction and the very central role of Ca^{2+}-ions as the ubiquitous "second messenger" in cells [1]. Intracellular changes in Ca^{2+}-ion concentration are involved in many important cellular events, for example, neurotransmitter release, or the excitation-contraction coupling in heart and skeletal muscle.

The power of modern fluorescence imaging techniques has allowed the monitoring of processes from the cellular down to the molecular level. As an example, measurements of the overall cellular Ca^{2+}-level

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

have contributed significantly to our knowledge about changes in intracellular *calcium* concentration under diseased conditions for many pathogenic diseases (e. g., [2]). Ca^{2+}-waves have been identified as complex spatio-temporal phenomena inherent to all excitable cells (e. g., [3]). But also the fundamental molecular events of Ca^{2+}-release, the Ca^{2+}-sparks, could be detected in heart and skeletal muscle with fast confocal scanning fluorescence microscopic techniques [4, 5, 6]. Thus, it is of fundamental importance to have a detailed and accurate analysis of fluorescence images in order to derive the cellular or subcellular Ca^{2+}-ion distribution with the highest possible accuracy. Potential pitfalls using *fluorescence indicators* are the limited kinetic properties of the dye and also the interaction of free Ca^{2+}-ions with a variety of buffers present in the system.

The most powerful method to handle these problems is a *model-based analysis* of the fluorescence images, as mathematical models provide the unbiased concentration profiles of any substance of interest. Examples where this method recently has been successfully applied are models of biphasic Ca^{2+}-transients in electrically nonexcitable cells [7], the analysis of Ca^{2+}-waves [8], and the analysis of Ca^{2+}-sparks [9].

In this chapter we will discuss the method of a model-based analysis of fluorescence images on the basis of Ca^{2+}-transients from muscle fiber preparations, which serve as good examples for the function of cellular systems due to their complex internal structure.

34.2 The necessity of modeling fluorescence images

When using *fluorescence imaging* data as a quantitative measure of intracellular ion concentrations, several problems arise. For Ca^{2+}-ion determination the most important issues are:

- Ca^{2+}-ions inside the cell are present in various different “states” (Fig. 34.1). In addition to the free ions, calcium is also bound to several intrinsic and extrinsic Ca^{2+}-binding sites (of which the fluorescence indicator is very often the most important one). Ca^{2+}-ions can also be sequestered in intracellular membrane-bound organelles, for example, mitochondria, the endoplasmic reticulum (ER), or the sarcoplasmic reticulum (SR) of muscle fibers;
- Fluorescence indicators generally affect the free ion concentrations by their own binding affinity. Also, the fluorescent dye will follow changes in free ion concentrations only with a more or less pronounced delay due to inherent kinetic limitations of the buffer-ion interaction;
- Very often the distribution of the fluorescent dye inside the cell is not known in detail, for example, the dye can also cluster on the sur-

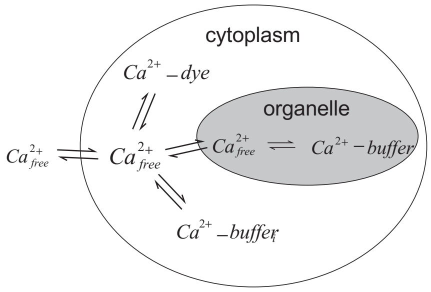


Figure 34.1: *The various states of calcium in a cell and their interplay.*

face or inside membrane-bound organelles, and therefore the origination of the measured fluorescence signal is often unclear;

- The different states of Ca^{2+}-ions interact by very complex processes: Ca^{2+}-ions are released or actively transported by cellular proteins, the binding of Ca^{2+}-ions is kinetically limited by the on- and off-rate constants of each buffer (this is a very important issue especially when measuring fast concentration changes with fluorescent dyes), and all mobile components of the cellular system also undergo diffusional translocation;
- Many quantities of biophysical and physiological relevance are not directly accessible with measurements, for example, release fluxes of Ca^{2+}-ions from intracellular stores, or the total Ca^{2+}-ion concentration. Failures in the regulation of the total Ca^{2+}-ion concentration are very often involved in the pathogenesis of severe diseases, for example, in Duchenne muscular dystrophy [2]; and
- The contribution of a single process to transient changes in the intracellular Ca^{2+}-concentration is very hard to determine from measurements of the complex system alone. The detailed knowledge of the composition of the macroscopic Ca^{2+}-change from individual Ca^{2+}-influencing processes is especially important when failures of cellular constituents are thought to be involved in pathogenic cases.

From the forementioned points it should be clear that fluorescence measurements alone will certainly miss very sharp intracellular spatial gradients and that the time course of very fast changes in Ca^{2+}-ion concentration cannot directly be monitored via a change in the detected fluorescence signal. In the following we will therefore discuss the method of modeling fluorescent images to get a better understanding of the true spatiotemporal concentration distribution of Ca^{2+}-ions in muscle fibers.

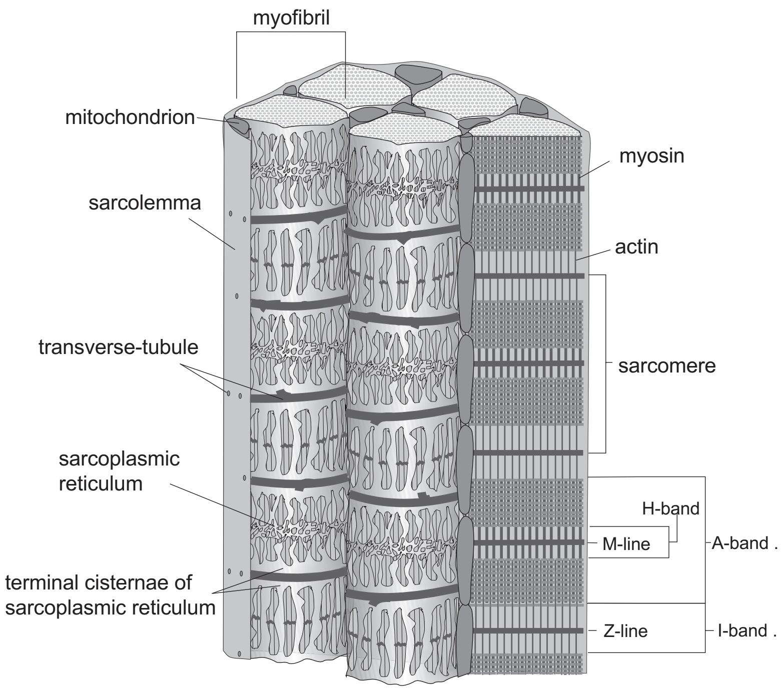


Figure 34.2: *The complex structure of a skeletal muscle fiber. The myofibrils are surrounded by the membranous network of the sarcoplasmic reticulum, which serves as the main reservoir for Ca^{2+}-ions sequestered from the cytosol. [Figure courtesy of Uwe Knobloch, Heidelberg.]*

34.3 The complex structure of muscle cells

Figure 34.2 is an example of the complexity of differentiated cells, which are far more complex than approximated in the sketch of Fig. 34.1. Skeletal muscle fibers are approximately cylindrical in shape and the smallest contractile units, the so-called myofibrils, contain the contractile proteins actin and myosin arranged in parallel filament systems, which can slide past each other during contraction. The force necessary for the contraction is the result of the conformational change in the myosin heads interacting with the actin filament. The contraction is regulated by the intracellular Ca^{2+}-concentration, which is mainly controlled by the membranous network of the sarcoplasmic reticulum (SR), the intracellular Ca^{2+}-ion store.

34.3.1 Ca^{2+}-regulation of muscular contraction

In general, the fast activation of skeletal muscle within a few milliseconds is initiated by an action potential generated by a nerve impulse, which is transmitted to the interior of the muscle fiber via the transverse tubular system leading to the voltage-dependent release of Ca^{2+}-ions from the SR. These ions spread in the myoplasm by diffusion and initiate a transient increase in the free myoplasmic Ca^{2+}-concentration by one to two orders of magnitude leading via activation of the Ca^{2+}-binding troponin/tropomyosin complex to the contraction of the muscle fiber.

There is still very little known about the later stages of Ca^{2+}-uptake by the SR, which is essential for the contraction-relaxation process. Apart from binding to intracellular buffers (e. g., parvalbumins, troponin-C), Ca^{2+}-ions are actively removed from the myoplasm by the powerful Ca^{2+}-ATPase, which actively pumps Ca^{2+}-ions back into the SR and thus lowers the intracellular Ca^{2+}-concentration to achieve relaxation.

34.3.2 Skinned muscle fibers

The function of the SR and its constituents, for example, the Ca^{2+}-ATPase, can be extensively studied in “skinned fiber” preparations, where the outer sarcolemma has been removed either by mechanical microdissection, by UV-laser microdissection or by chemical means [10], thus allowing direct diffusional access to the myoplasm. Effects of various drugs on the contractile proteins and on components of the SR can directly be investigated by recording caffeine-induced force and fluorescence transients, which are well-established indicators to monitor the Ca^{2+}-handling in these preparations.

34.4 Ca^{2+}-measurements

The fluorescence data of the caffeine-induced Ca^{2+}-release in Fig. 34.3 is recorded with a combined photometric and imaging system (for details see [11]) using the ratiometric Ca^{2+}-sensitive dye *Fura-2* [12]. Fura-2 allows *dual excitation* measurements and thereby offers the advantage that the fluorescence signal is independent of critical quantities, for example, the dye concentration and the specimen geometry. Therefore the changes in the fluorescence ratio signal can be directly correlated with a change in the Ca^{2+}-ion concentration bound to the fluorescent indicator.

Figure 34.3 is a typical example of a caffeine-induced Ca^{2+}-transient in skinned muscle fiber preparations recorded with Fura-2. After the addition of caffeine at time $t = 0\,\mathrm{s}$ the release of Ca^{2+}-ions inside the muscle fiber for times $t < 2.4\,\mathrm{s}$ can be seen and for times $t > 2.4\,\mathrm{s}$ the

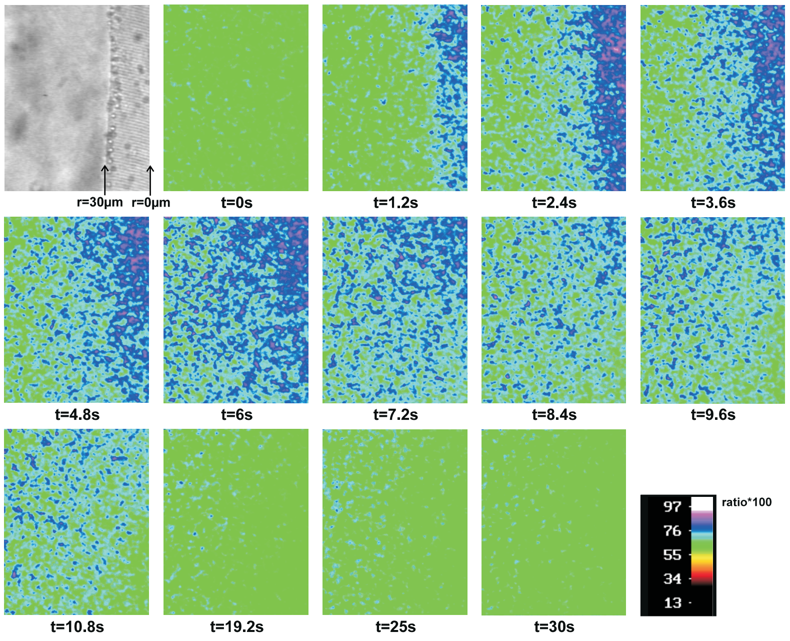


Figure 34.3: *Time series of a caffeine-induced Ca^{2+}-transient recorded with the Ca^{2+}-sensitive dye Fura-2 (10 μM) using the wavelength pair 340 nm/380 nm [taken from Uttenweiler et al. [13]]; (see also Plate 9).*

contribution of Ca^{2+}-diffusion out of the fiber through the permeabilized sarcolemma is clearly visible. Due to the diffusional loss of Ca^{2+}-ions the radial concentration profile is of very special interest in these experiments. Therefore intensity cuts perpendicular to the muscle fiber axis were acquired from each ratio image of the time series. The alignment of the radial profiles in time leads to the graph of Fig. 34.4, which consequently reflects the spatiotemporal Ca^{2+}-ion distribution.

However, for the exact interpretation of the experimental findings further analysis is required. The solution surrounding the muscle fibers in these experiments contains large amounts of extrinsic buffers, which comprises 0.5 mM EGTA and the fluorescence indicator Fura-2 itself. Therefore the accurate description of the total Ca^{2+}-turnover can only be achieved by using a more sophisticated analysis of the fluorescent images.

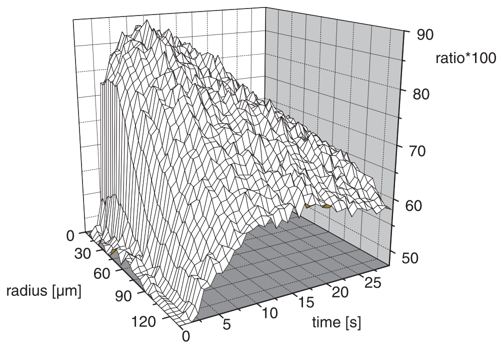


Figure 34.4: *Radial concentration profile of Ca^{2+}-ions as obtained from intensity cuts perpendicular to the muscle fiber axis of each ratio image of the time series in Fig. 34.3. The muscle fiber center is at $r = 0\,\mu m$ and the surface at $r = 30\,\mu m$.*

34.5 Mathematical modeling

Mathematical models of cellular and subcellular systems mostly have to use numerical methods as these systems are far too complex to be treated analytically. These models often have to take advantage of some kind of inherent symmetry to allow an easier formulation of the associated differential equations and to reduce computational load. Spherical symmetry is frequently assumed for modeling neurons (e. g., [14]) and models of muscle fibers can mostly use cylindrical symmetry (e. g., [13, 15]). The biophysical processes involved in cellular functions can be described by partial differential equations and the respective numerical solutions can be obtained using a suitable discrete grid (see [16]).

In the following, a discussion of a numerical model using an explicit finite difference approximation will be given, which attempts to describe the Ca^{2+}-turnover in skinned muscle fibers, as measured in the fluorescence experiments presented in the previous section.

34.5.1 Numerical model of the Ca^{2+}-turnover

As skeletal muscle fibers have approximately cylindrical shape mathematical models can exploit the cylindrical geometry for an easier formulation of the equations associated with each process. All model calculations were performed assuming homogeneity along the fiber axis and radial symmetry and the experimental geometry is approximated as shown in Fig. 34.5.

As the main process of Ca^{2+}-translocation is diffusion, modeling the Ca^{2+}-transient leads to the solution of the *diffusion equation* with

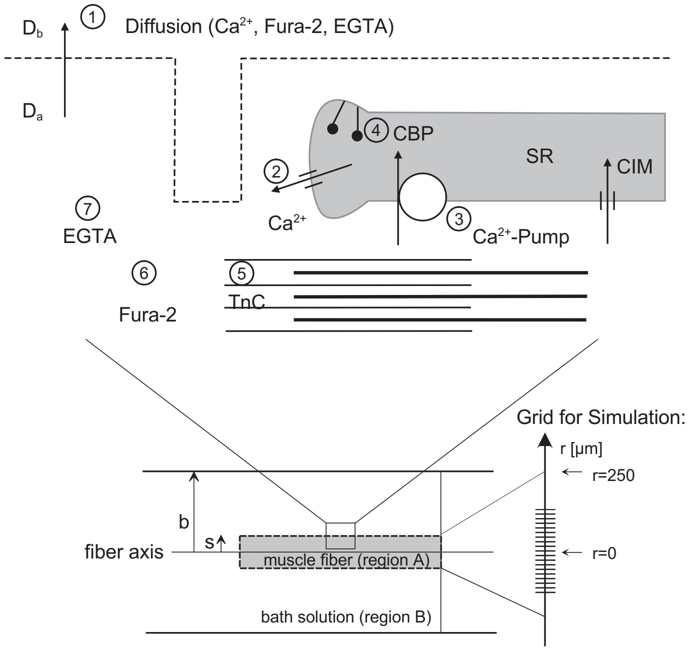


Figure 34.5: *Model of a skinned muscle fiber assuming cylindrical symmetry. The different components are: diffusion of free and bound Ca^{2+}-ions as well as the diffusion of mobile buffers (EGTA, Fura-2), binding to mobile and immobile buffers, Ca^{2+}-release and reuptake by the SR and binding of Ca^{2+}-ions to Ca^{2+}-binding proteins inside the SR.*

various sink and source terms. The diffusion equation in cylindrical coordinates where diffusion occurs purely in radial direction can be written as

$$\frac{\partial c(r,t)}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(rD\frac{\partial c(r,t)}{\partial r}\right) + h(r,t) \tag{34.1}$$

with

$$h(r,t) = \sum_l h_l(r,t)$$

where c is the concentration, D is the *diffusion coefficient* of the diffusing substance, r is the radial coordinate, t is the time coordinate, and $h(r,t)$ is the sum of all source and sink terms $h_l(r,t)$ of the various processes involved.

Let [Ca^{2+}] denote the free Ca^{2+}-concentration in the following equations.

The release of Ca^{2+}-ions from the sarcoplasmic reticulum is assumed to be proportional to the concentration gradient across the SR membrane

$$\frac{d[\mathrm{Ca}^{2+}]}{dt} = k_1 \left([\mathrm{Ca}^{2+}]_{SR} - [\mathrm{Ca}^{2+}]_{\mathrm{myoplasm}}\right) \tag{34.2}$$

where k_1 is the proportional constant, which can be used to adjust the extent of SR Ca^{2+}-ion release per unit time.

The active removal of Ca^{2+}-ions from the cytosol by the SR Ca^{2+}-pump is modeled with a Hill-type relation, assuming a Ca^{2+}-dependent second-order saturable pump. The uptake of Ca^{2+}-ions into the SR can then be written as

$$\frac{d[\mathrm{Ca}^{2+}]}{dt} = p\frac{v_{\max}[\mathrm{Ca}^{2+}]^n}{[\mathrm{Ca}^{2+}]^n + K_m^n} \tag{34.3}$$

where $v_{\max}$ is the maximum uptake velocity, K_m is the half-maximal uptake rate, $n = 2$ is the Hill-coefficient, and p is the proportional factor.

Calcium is assumed to bind to all buffers in a 1:1 stoichiometry and without cooperativity, so that the following equation holds:

$$\frac{d[\mathrm{Ca}^{2+}]}{dt} = k_{\mathrm{on}}^l[\mathrm{Ca}^{2+}]_{\mathrm{free}} \cdot [\mathrm{buffer}^l]_{\mathrm{free}} - k_{\mathrm{off}}^l \cdot [\mathrm{Ca}^{2+} - \mathrm{buffer}^l] \tag{34.4}$$

where $l = 3, 4, 5$ is the index for the various buffers, k_{on}^l is the kinetic on-rate constant, k_{off}^l is the kinetic off-rate constant of the $\mathrm{buffer}^l-\mathrm{Ca}^{2+}$ binding.

The finite difference approximation of the diffusion equation (34.1) without sink and source terms neglecting the error term is given by Crank [17]

$$\begin{aligned} c_{i,j+1} &= \frac{D}{2i}\frac{\Delta t}{(\Delta r)^2}\left[(2i+1)c_{i+1,j} - 4ic_{i,j} + (2i-1)c_{i-1,j}\right] + c_{i,j} \\ c_{0,j+1} &= 4D\frac{\Delta t}{(\Delta r)^2}(c_{1,j} - c_{0,j}) + c_{0,j} \end{aligned} \tag{34.5}$$

The upper equation is for $i \neq 0$ only. The indices i and j denote the radial grid position and the discrete time index, respectively. Similar finite difference formulae can be found for the other differential Eqs. (34.2) to (34.4).

The boundaries of the system have to be treated with reasonable boundary conditions. In the present case Neumann boundary conditions naturally apply, as there can be no flux perpendicular to the glass surface of the experimental chamber. Also, the interface between two different media, in this case between cytosol and surrounding bath solution, has in general to be treated separately (for details see [13]). The rate equation for the change in free myoplasmic calcium concentration at grid points inside the fiber volume is given by

$$c_{i,j+1} = \mathit{diff} + h_{1_{i,j}} - h_{2_{i,j}} - h_{3_{i,j}} - h_{4_{i,j}} - h_{5_{i,j}} \tag{34.6}$$

where *diff* stands for the finite difference formula described by Eq. (34.5) for diffusion, $h_{1_{i,j}}$ is the Ca^{2+}-release term, $h_{2_{i,j}}$ the Ca^{2+}-pump term, $h_{3_{i,j}}$ is the buffering of troponin-C, $h_{4_{i,j}}$ the buffering of EGTA and $h_{5_{i,j}}$ the buffering of Fura-2. Similar rate equations can be obtained for the concentration of each substance and for the grid points outside the fiber volume. Finally, the rate equation for the free Ca^{2+}-concentration inside the sarcoplasmic reticulum can be written as

$$c^{SR}_{i,j+1} = v_{SR}\left(h_{2_{i,j}} - h_{1_{i,j}}\right) - h_{6_{i,j}} + c^{SR}_{i,j} \qquad (34.7)$$

where $h_{6_{i,j}}$ denotes calsequestrin buffering and v_{SR} =10 is the volume factor compensating the fact that the SR occupies only 10 % of the fiber volume.

34.5.2 Parameters

All detailed mathematical models have a large set of input parameters, most of which are very often only poorly known under the given experimental conditions. Therefore they frequently have to be chosen from within a broader range of values and many parameters have to be adjusted to obtain an optimum fit between the simulations and the experiments.

The diffusion coefficients of calcium and all mobile buffers are significantly different in free solution and in the cytosol. The diffusion coefficients are only roughly half the value in the cytosol (for free Ca^{2+}-ions: D$_{\text{cytosol}}$ ≈ 225-300 μm^{2}s^{-1}, D$_{\text{free}}$ ≈ 700 μm^{2}s^{-1}), thus, all models have to include the distinction between the different media. Also, the concept of effective diffusion coefficients has been introduced [18]. The kinetic on- and off-rate constants of many Ca^{2+}-buffers have been measured *in vivo* and *in vitro*, but uncertainties still originate from the fact that these quantities are highly sensitive on experimental conditions, for example, ionic strength and pH. Therefore, corrections regarding the experimental conditions are very often required.

The various parameters for the simulation presented were taken from Grynkiewicz et al. [12], Wagner and Keizer [18], Cannell and Allen [19], Donoso et al. [20], Pape et al. [21], Stienen et al. [22], and Robertson et al. [23] as detailed in Uttenweiler et al. [13].

34.6 Simulated Ca^{2+}-transients

The radial Ca^{2+}-ion distribution as calculated from the model is shown in Fig. 34.6. For the details, especially the calibration, see Uttenweiler et al. [13]. It is evident that the radial Ca^{2+}-Fura-2 distribution roughly corresponds to the measured radial Fura-2 fluorescence signal. The

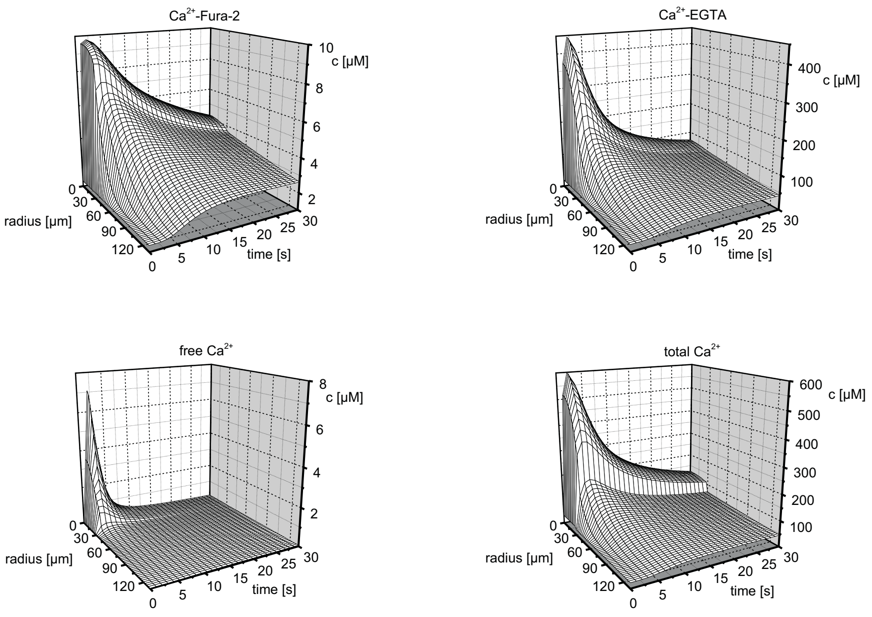


Figure 34.6: *Modeled radial concentration distribution for the Ca^{2+}-Fura-2 complex, the Ca^{2+}-EGTA complex and for the free Ca^{2+}-ions and the total calcium.*

other concentrations of interest, for example, the free Ca^{2+}-ion distribution, are significantly different from the measured fluorescence signal. This can not solely be explained by the nonlinear calibration curve of the fluorescent dye.

The differences mainly arise from the kinetic properties of the fluorescence dye and the high extraneous buffer capacities. In general, the error due to the kinetic limitations of the fluorescence indicator is larger, the faster are the processes under consideration. Spatial gradients are also mostly underestimated [15] and therefore, in general, the true spatiotemporal Ca^{2+}-ion distribution in fluorescence images has to be calculated with mathematical models.

Mathematical models also offer the advantage that quantities, which are normally not accessible with measurements, can be studied. The reconstruction of the total Ca^{2+}-ion distribution in Fig. 34.6 and the various concentrations and fluxes shown in Fig. 34.7 are some examples. Very often not only the knowledge of the free Ca^{2+}-concentration but also of other quantities is of central importance and therefore a model-based analysis of fluorescence imaging data can yield new insight into previously unknown processes.

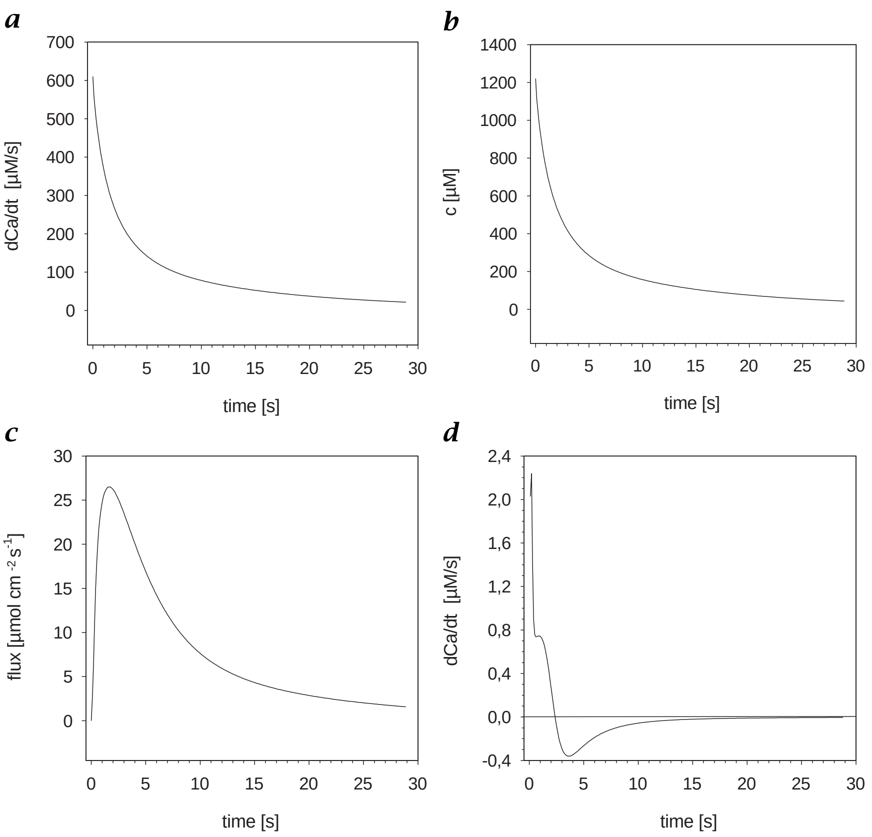


Figure 34.7: *Examples of quantities derived from the model:* ***a*** *time course of the release rate of Ca^{2+}-ions from the sarcoplasmic reticulum;* ***b*** *time course of the free Ca^{2+}-ion concentration inside the sarcoplasmic reticulum;* ***c*** *flux of Ca^{2+}-ions across the permeabilized sarcolemma from the fiber preparation into the surrounding bath solution;* ***d*** *time course of the rate of change in free Ca^{2+}-ion concentration inside the fiber preparation.*

34.7 Conclusions

The method of a model-based analysis of fluorescence images provides a very powerful tool to study intracellular ion concentrations in general. Especially, the investigation of the regulation of the intracellular Ca^{2+}-concentration by the SR in muscle fibers requires a very precise understanding of the spatio-temporal Ca^{2+}-distribution.

Fluorescence images acquired with various techniques (see Chapters 21 and 12) yield information about the distribution of Ca^{2+}-ions bound to the fluorescence indicator. But only mathematical models, which account for the kinetic properties of the dye and buffering effects of all buffer systems present, will reveal the unbiased distribution of free Ca^{2+}-ions and all other information that is explicitly or implicitly present in the model.

The availability of quantities normally inaccessible by measurements is a large advantage, as they can be very sensitive measures for pathogenic changes in functional components of intracellular structures and thereby help to expand the experimental studies on severe diseases.

In the future a direct model-based analysis of fluorescence images will certainly be available, where the information derived from model calculations can already be used for the analysis of the image sequence.

This approach generally is very useful for all scientific applications where fluorescence indicators are used, as the kinetic limitations of fluorescence dyes and the buffering of ions are commonly encountered problems.

34.8 References

[1] Thomas, M. V., (1982). *Techniques in Calcium Research*. London: Academic Press.

[2] Gailly, P., Boland, B., Himpens, B., Casteels, R., and Gillis, J. M., (1993). Critical evaluation of cytosolic calcium determination in resting muscle fibres from normal and dystrophic (mdx) mice. *Cell Calcium*, **14**:473–483.

[3] Wussling, M. H. P. and Salz, H., (1996). Nonlinear propagation of spherical calcium waves in rat cardiac myocytes. *Biophys. J.*, **70**:1144–1153.

[4] Klein, M. G., Cheng, H., Santana, L. F., Jiang, Y.-H., Lederer, W. J., and Schneider, M. F., (1996). Two mechanisms of quantized calcium release in skeletal muscle. *Nature*, **379**:455–458.

[5] Lipp, P. and Niggli, E., (1996). Submicroscopic calcium signals as fundamental events of excitation contraction coupling in guinea pig cardiac myocytes. *J. Physiol.*, **492**:31–38.

[6] Tsugorka, A., Rios, E., and Blatter, L., (1995). Imaging elementary events of calcium release in skeletal muscle cells. *Science*, **269**:1723–1726.

[7] Korngreen, A., Goldstein, V., and Priel, Z., (1997). A realistic model of biphasic calcium transients in electrically nonexcitable cells. *Biophys. J.*, **73**:659–673.

[8] Kupferman, R., Mitra, P. P., Hohenberg, P. C., and S.-H-Wang, S., (1997). Analytical calculation of intracellular calcium wave characteristics. *Biophys. J.*, **72**:2430–2444.

[9] Cannel, M. B. and Soeller, C., (1997). Numerical analysis of ryanodine receptor activation by L-type channel activity in the cardiac muscle diad. *Biophys. J.*, **73**:112–122.

[10] Veigel, C., Wiegand-Steubing, R., Harim, A., Weber, C., Greulich, K. O., and Fink, R. H. A., (1994). New cell biological applications of the laser microbeam technique: the microdissection and skinning of muscle fibres

and the perforation and fusion of sarcolemma vesicles. *European Jour. Cell Biology*, **63**:140–148.

[11] Uttenweiler, D., Wojciechowski, R., Makabe, M., Veigel, C., and Fink, R. H. A., (1995). Combined analysis of intracellular calcium with dual excitation fluorescence photometry and imaging. *Optical Engineering*, **34(10)**: 2864–2871.

[12] Grynkiewicz, G., Poenie, M., and Tsien, R. Y., (1985). A new generation of Ca^{2+} indicators with greatly improved fluorescence properties. *The Journal of Biological Chemistry*, **260**:3440–3450.

[13] Uttenweiler, D., Weber, C., and Fink, R. H. A., (1998). Mathematical modeling and fluorescence imaging to study the Ca^{2+}-turnover in skinned muscle fibers. *Biophys. J.*, **74(4)**:1640–1653.

[14] Sala, F. and Hernandez-Cruz, A., (1990). Calcium diffusion modeling in a spherical neuron. *Biophys. J.*, **57**:313–324.

[15] Duty, S. and Allen, D. G., (1994). The distribution of intracellular calcium concentration in isolated single fibres of mouse skeletal muscle during fatiguing stimulation. *Pflügers Arch.*, **427**:102–109.

[16] Ames, W. F., (1984). *Numerical Methods for Partial Differential Equations.* Boston: Academic Press.

[17] Crank, J., (1975). *The Mathematics of Diffusion.* 2nd edition. Oxford: Oxford University Press.

[18] Wagner, J. and Keizer, J., (1994). Effects of rapid buffers on Ca^{2+} diffusion and Ca^{2+} oscillations. *Biophys. J.*, **67**:447–456.

[19] Cannell, M. B. and Allen, D. G., (1984). Model of calcium movement during activation in the sarcomere of frog skeletal muscle. *Biophys. J.*, **45**:913–925.

[20] Donoso, P., Prieto, H., and Hidalgo, C., (1995). Luminal calcium regulates calcium release in triads isolated from frog and rabbit skeletal muscle. *Biophys. J.*, **68(2)**:507–515.

[21] Pape, P. C., Jong, D. S., and Chandler, W. K., (1995). Calcium release and its voltage dependence in frog cut muscle fibres equilibrated with 20 mM EGTA. *J. Gen. Physiol.*, **106**:259–336.

[22] Stienen, G. J. M., Zaremba, R., and Elzinga, G., (1995). ATP utilization for calcium uptake and force production in skinned muscle fibres of *Xenopus laevis*. *J. Physiol.*, **482(1)**:109–122.

[23] Robertson, S. P., Johnson, J. D., and Potter, J. D., (1981). The time-course of Ca^{2+} exchange with calmodulin, troponin, parvalbumin, and myosin in response to transient increases in Ca^{2+}. *Biophys. J.*, **34**:554–569.

35 Studies of Small-Scale Air-Sea Interaction with Active and Passive Thermography

Uwe Schimpf[1,2], Horst Haußecker[2], and Bernd Jähne[1,2]

[1]Scripps Institution of Oceanography, La Jolla, USA
[2]Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR)
Universität Heidelberg, Germany

35.1 Introduction

In recent years, *global climate change* and growing atmospheric and oceanic pollution have drawn the attention of scientific research to the global cycles in the environment. Accelerated burning of fossil fuels and the diminishing tropical forests have led to a threatening increase of atmospheric carbon dioxide, which may have an important impact on the earth's climate. Since the Industrial Revolution, the concentration of carbon dioxide in the atmosphere increased by 25 %. How much does human influence contribute to the global CO_2 cycle and global warming, the greenhouse effect, and weather phenomena, such as El Niño?

The oceans cover two-thirds of the earth's surface. Driven by the wind and heated by solar radiation, this moving bulk of water directly affects the climate. These masses of flowing water have an enormous capacity to absorb and release both heat and gases. About 7 Gt[1] carbon is released yearly into the atmosphere [1]. As already mentioned,

[1] 1 Gt=10^{15} g

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

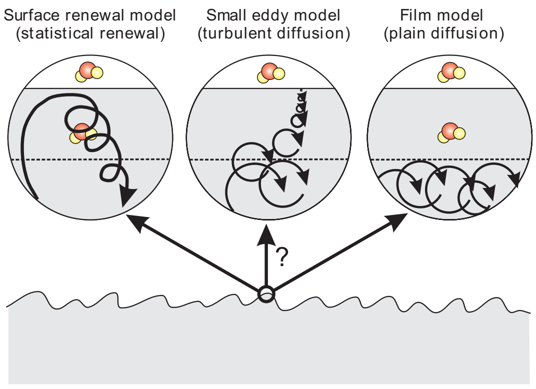

Surface renewal model (statistical renewal)
Small eddy model (turbulent diffusion)
Film model (plain diffusion)
?

Figure 35.1: *Which of the three simple turbulence models underlies the transport process across the air/water interface: statistical renewal, turbulent diffusion, or plain diffusion?*

the oceans play an important role as a carbon sink. Investigations by Keeling and Shertz [2] estimate the yearly absorption capacity of the oceans to 3.0±2.0 Gt carbon.

The development of climatological models requires an understanding of the various parameters that influence the transfer of heat and gas across the air-sea interface [3]. The knowledge of the underlying processes is quite incomplete as the transfer across the air-sea interface is very difficult to observe and is dominated by the first millimeter of the ocean surface [4]. Which of the three simple turbulence models underlies the transport process across the aqueous boundary layer at the air/water interface (see Fig. 35.1): statistical renewal, turbulent diffusion, or plain diffusion?

In recent years, new image processing techniques have been developed to obtain an insight into these processes in the laboratory and the field [5]. Visualization and quantitative investigation of gas and heat exchange have become feasible due to rapid progress in computer hardware, availability of high sensitive cameras, and the development of new algorithms.

35.2 The controlled flux technique

The *Controlled Flux Technique* (CFT) [6] uses heat as a proxy tracer for gases to measure the air-sea gas transfer velocity. In contrast to conventional techniques based on mass balances, the CFT allows the transfer velocity to be estimated locally and with a temporal resolution of less than a minute. Additionally, image processing techniques allow

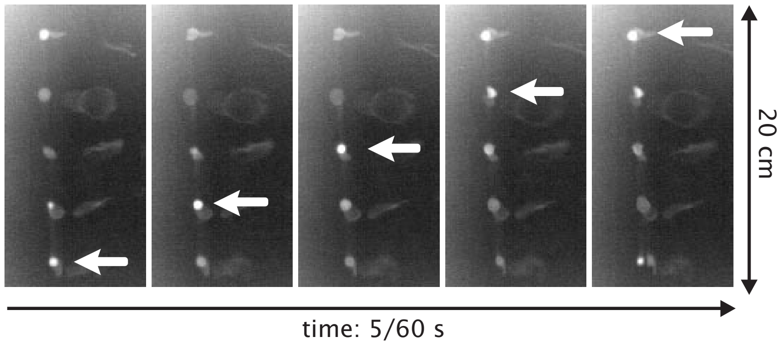


Figure 35.2: *Active thermography: small patches at the water surface are heated up periodically by an infrared CO_2-laser. The spots in the infrared image sequences are tracked and the gas exchange rate is calculated from their temporal decay; for movie see* /movies/35/active.mov.

for a detailed investigation of the flow field and the microturbulence at the ocean surface, which are important parameters of the transport mechanisms.

Conceptually speaking, the transfer velocity k (defined in Eq. (35.1)) is a piston velocity, which is an averaged measure of how fast a gas or heat penetrates from the airside of the interface into the water. The transfer velocity relates the flux j across the interface to the concentration difference. As the transport mechanisms for gas and heat are governed by the same physical laws (diffusion and convection), heat can be used as a proxy tracer for gases.

The principle of this new technique is very simple. Infrared radiation is used to heat up the water surface and from the resulting temperature response the transfer velocity k for heat in water is determined by:

$$k = \frac{j}{\Delta C} = \frac{j}{\rho c_v \Delta T} = \sqrt{\frac{D_h}{t_*}} \qquad (35.1)$$

where j denotes the flux at the water surface, ΔC the concentration difference, ρ the density of water, c_v the specific heat capacity of water, ΔT the temperature difference across the aqueous boundary layer, D_h the diffusion coefficient for heat in water, and t_* the time constant for the transfer process.

Two different techniques, an active and a passive method, are applied to measure the transfer velocity for heat. Both methods use image processing techniques and yield independent estimates of the transfer velocity and, thus, can be used to cross verify each other.

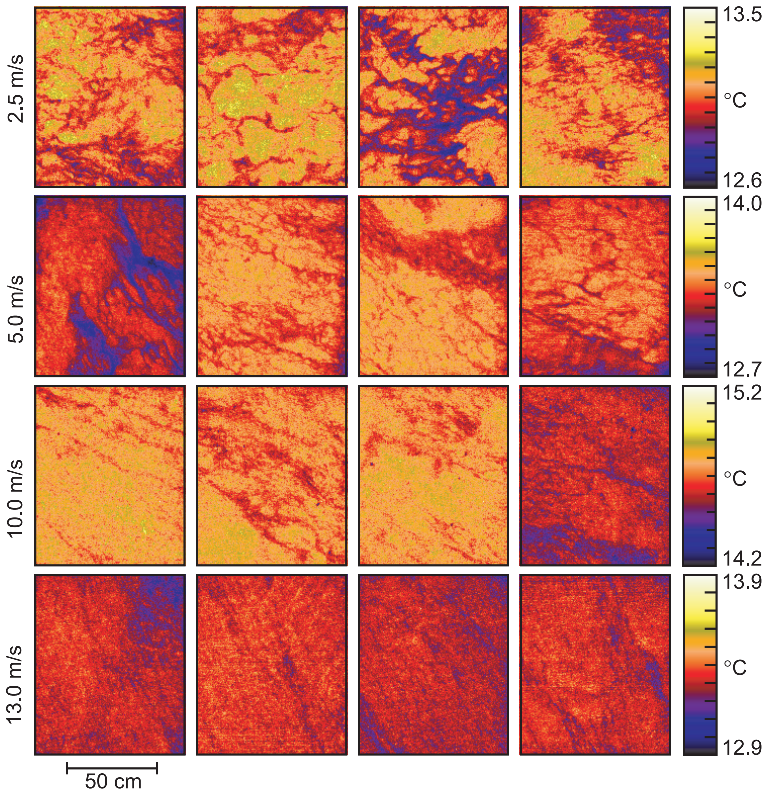


Figure 35.3: *Infrared images of the water surface at different wind speeds. The image brightness is temperature calibrated. Obviously, the temperature contrast and the size of the structures on the water surface decrease with higher wind speeds; for movies see* `/movies/35/p329.mov`, `/movies/35/p330.mov`, *and* `/movies/35/p331.mov`; *(see also Plate 10).*

Using *active thermography*, small patches at the water surface are heated up periodically by an infrared laser (see Fig. 35.2). The infrared camera system visualizes the temporal decay process of the temperature at the water surface with a resolution of up to 120 frames/s. Within the recorded image sequences, the heat spots are tracked using a tensor approach for low-level motion estimation [Volume 2, Section 13.3.2]. The obtained decay curves of the temperature distribution yield the time constant t_* of the heat transfer. According to the general equation of the gas exchange Eq. (35.1), the transfer velocity k is calculated from t_* (compare Fig. 35.4).

Applying *passive thermography*, no artificial heating is necessary. The infrared camera is sensitive enough (compare Section 35.4) to visualize even minor temperature fluctuations at the ocean surface (see

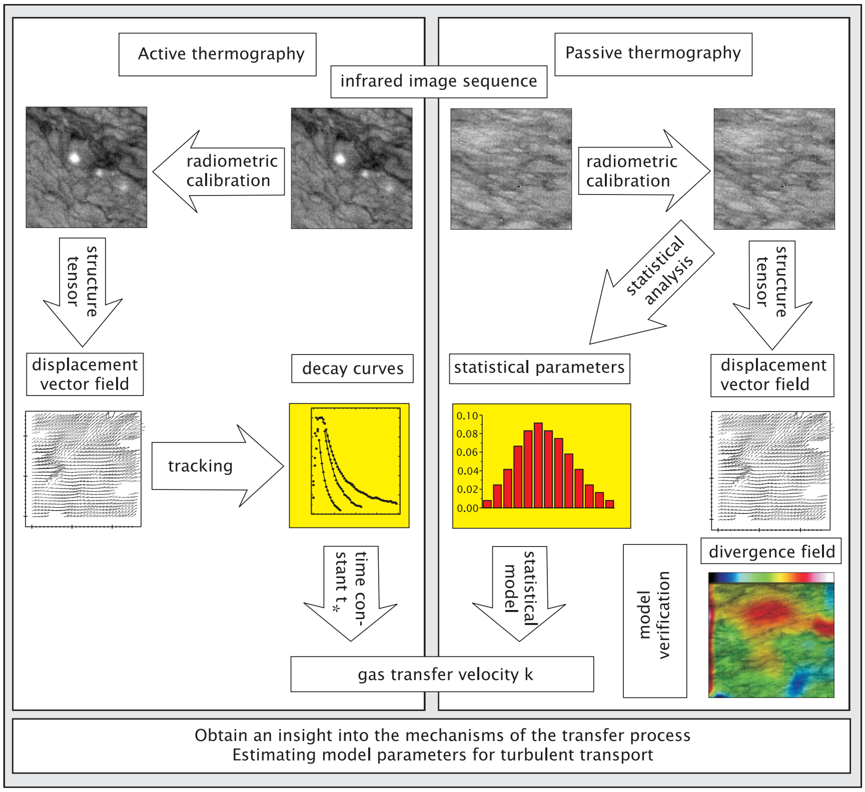


Figure 35.4: *Schematic overview of the image processing techniques used to investigate the gas transfer process across the air/sea interface with infrared imagery.*

Fig. 35.3). Due to latent and sensible heat fluxes and radiative cooling at the ocean surface, the ocean skin is generally a few tenths of a degree colder than the water bulk. This phenomenon is commonly referred to as "*cool skin of the ocean*." The temperature difference ΔT across the interface is directly estimated from statistical properties of the natural *sea surface temperature* (SST) obtained from the infrared image sequences. Once the temperature difference is computed, the transfer velocity k is determined by Eq. (35.1). An accurate calibration technique (Section 35.4) is required to get reliable temperature information from the infrared imaging system.

In image sequences of these temperature fluctuations, details of the microturbulence of the water surface become visible. This discovery provides the unique opportunity of determining the 2-D flow field and the turbulence statistics at the ocean surface from infrared image sequences [7]. Figure 35.4 shows an overview of the used image process-

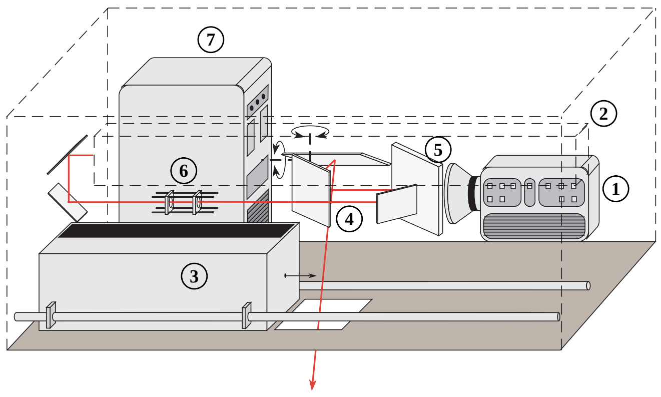


Figure 35.5: *Schematic setup of the CFT. 1: infrared camera; 2: CO_2-laser; 3: calibration device; 4: x/y-scanner; 5: beam splitter; 6: laser optic; 7: PC.*

ing techniques. The 2-D flow field was computed using tensor-based techniques detailed in Volume 2, Section 13.3.2. From the flow field, higher order properties, such as the divergence and vorticity can be calculated. These results give a direct insight into the physics of the transfer process [8, 9].

35.3 Experimental setup

The main part of the CFT system combines an infrared camera, an infrared CO_2-laser and a temperature calibration device. Figure 35.5 shows the schematic setup of the CFT instrument.

An infrared 25 W CO_2-laser, emitting radiation at a wavelength of 10.6 μm, heats up a small area of the water surface, either in continuous or pulsed mode. The AMBER Radiance 1 infrared camera has a 256×256 focal plane array (FPA), sensitive in the wavelength region from $3-5\,\mu$m with a $NE\Delta T$ (noise equivalent temperature difference) of only 0.02 K. Depending on the distance between the water surface and the instrument, the observed footprint varies from 0.3×0.3 m to 1.4×1.4 m.

A Gaussian telescope optic is used to adjust remotely the size of the laser spot from some millimeters up to some centimeters. With this feature, a large variety of heat patterns is drawn to the water surface (see Fig. 35.2). The temperature response as well as the spatial and temporal behavior of the heat patterns are investigated as described in Section 35.2.

Two personal computers are used to control all components of the instrument and to transfer and store the image sequences and analog data. One of the two computers, located in the instrument box, is

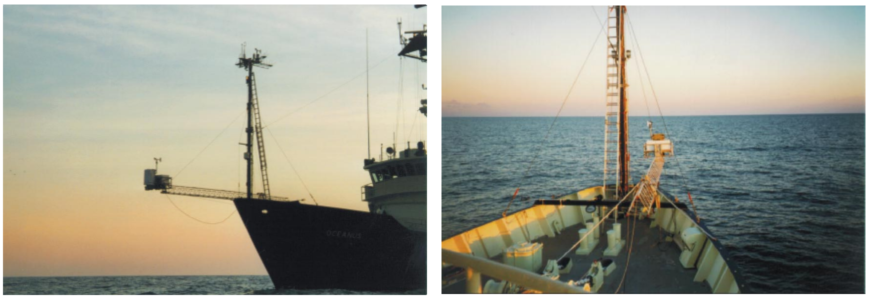


Figure 35.6: *The CFT instrument is mounted on a 7-m long boom at the bow of the Research Vessel Oceanus during the cruise in the North Atlantic, July 1997; (see also Plate 11).*

equipped with a PCI frame grabber [10] and a multifunction I/O-card [11]. The digital part of the I/O-card synchronizes the infrared CO_2-laser with the image acquisition board and controls the x/y-scanner [12] as well as the servos to focus the infrared camera and to move the calibration device into the optical path of the infrared camera. The analog module of the I/O-board acquires temperature data from the PT-100 sensors, located in the calibration device (see Section 35.4).

In field campaigns, the CFT instrument was used during a cruise in the Pacific Ocean in the spring of 1995 and another one in the North Atlantic in the summer of 1997. The entire instrumentation was mounted on a boom at the bow of the ships (see Fig. 35.6). Several laboratory campaigns in the Delft Hydraulics Laboratory (Netherlands), in the Scripps Hydraulics Laboratory (U.S.), and in the Heidelberg wind-wave facility (Germany), took place to perform measurements under well-defined conditions.

35.4 Calibration

The infrared camera Amber Radiance I has a resolution of 12 Bit. To obtain a quantitative relation between the image brightness and the temperature, an accurate calibration has to be performed. For this purpose, a special calibration device was designed [13, 14]. An internal uniformity correction compensates the slight difference in the spatial sensitivity of the focal plane array [15]. Radiometric measurements [13] have shown that for an accurate quantitative analysis, this procedure is not sufficient and has to be replaced by a *Three-Point calibration process*.

For the calibration process, images from "black" surfaces with different temperatures are taken to obtain the characteristics of each sensor element from the focal plane array of the infrared camera. The

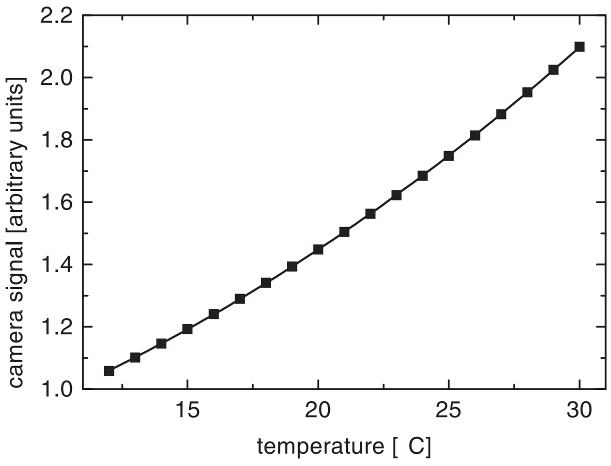


Figure 35.7: *The relation between the camera signal and the temperature $S(T)$ is well approximated by a second-order polynomial.*

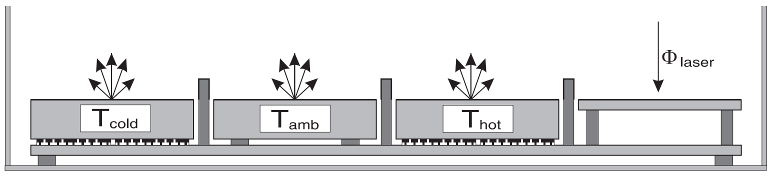


Figure 35.8: *Side view of the calibration device in the CFT instrument (schematic). The thinner aluminum body serves for the calibration of the infrared CO_2-laser as a heat sink.*

relation between the camera signal and the temperature S(T) can be well approximated by a second-order polynomial (Fig. 35.7).

Thus, a calibration device with three different temperature standards guarantees an accurate measurement of the temperature. Three thermally isolated aluminum bodies were brought to three different temperatures. Two Peltier elements are used to heat and cool two of the aluminum bodies, while the third one remains at ambient temperature (Fig. 35.8). The geometrical arrangement of these three temperature standards and the reference bodies reduce disturbing reflections up to 0.001 % of incident radiation.

The temperature of the aluminum bodies is measured with highly sensitive PT-100 elements. The relative sensitivity of these sensors is in the order of approximately 0.01 K, the absolute sensitivity approximately 0.1 K.

While recording images at different temperatures, the calibration curve S(T) is determined for any pixel. The reverse function T(S) delivers

Table 35.1: *Specifications of the developed calibration device for the infrared imaging system*

Specifications of the calibration device	
homogeneity of the temperature over the aperture	$\leq 0.025K$
absolute accuracy of temperature T	$0.1K$
relative accuracy of temperature ΔT	$0.025K$

the relation between gray value and absolute temperature. Let $S(T_1)$, $S(T_2)$, and $S(T_3)$ be the camera signal of the three aluminum bodies at the temperatures T_1, T_2, and T_3. The linear equation system:

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} 1 & S(T_1) & [S(T_1)]^2 \\ 1 & S(T_2) & [S(T_2)]^2 \\ 1 & S(T_3) & [S(T_3)]^2 \end{bmatrix} \begin{bmatrix} a_T \\ b_T \\ c_T \end{bmatrix} \qquad (35.2)$$

can be solved for the parameters a_T,b_T, and c_T of every sensor element by:

$$\begin{bmatrix} a_T \\ b_T \\ c_T \end{bmatrix} = \begin{bmatrix} 1 & S(T_1) & [S(T_1)]^2 \\ 1 & S(T_2) & [S(T_2)]^2 \\ 1 & S(T_3) & [S(T_3)]^2 \end{bmatrix}^{-1} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \qquad (35.3)$$

Using the estimated parameters from Eq. (35.3), the absolute temperature for every pixel of the sensor element is computed. The accuracy of temperature measurements using the calibration device in conjunction with the Amber Radiance 1 infrared camera is detailed in Table 35.1.

With the infrared camera and the calibration device, an infrared imaging system is available that allows us to distinguish relative temperature difference at the water surface of about 0.01 K and to determine absolute temperatures with an accuracy of 0.1 K.

35.5 Results and conclusions

As an example, results of field measurements during the spring 1995 cruise in the Pacific are shown in Fig. 35.9. The obtained transfer velocities from the CFT method and the wind speed are plotted versus time. The transfer rates are normalized to the transfer velocity of CO_2 at 20° and averaged over 4 min. These data points clearly demonstrate the necessity of high temporal resolved measurements of the transfer

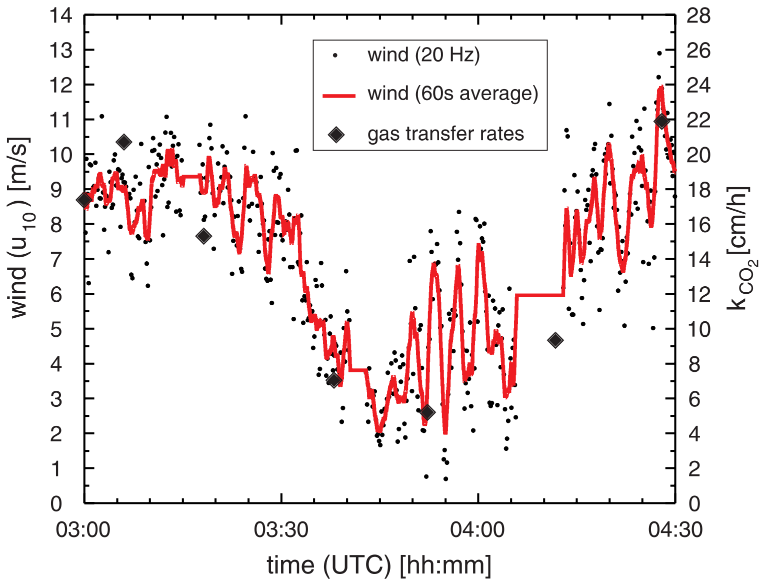


Figure 35.9: *Gas transfer rates and wind speeds in a time span of 90 min. The transfer velocities are normalized to the transfer velocity of CO_2 at 20° and averaged over 4 min.*

velocity. Within a time span of 90 min, the wind speed drops from 10 m/s to a calm breeze and picks up again up to 11 m/s. Conventional methods such as the mass balance method and dual tracer techniques have a time resolution of from several hours to days or weeks [3]. In this time period the meteorological conditions of the oceans are subject to changes and, thus, a parameterization of the transfer process with intermittent meteorological parameters is very difficult.

In Fig. 35.10, the transfer velocity k is plotted versus the wind speed. White symbols indicate field data of various authors from the past 20 yr [13], black dots indicate transfer velocities calculated from the controlled flux technique within a time span of several hours. These results show the enormous importance of this new approach for field measurements of gas exchange processes.

For the first time it is possible to determine gas exchange rates with a temporal resolution, which conforms to the time scale of intermittent meteorological conditions. Due to this fact, a physically based parameterization of the transfer process across the viscous boundary layer seems to be feasible.

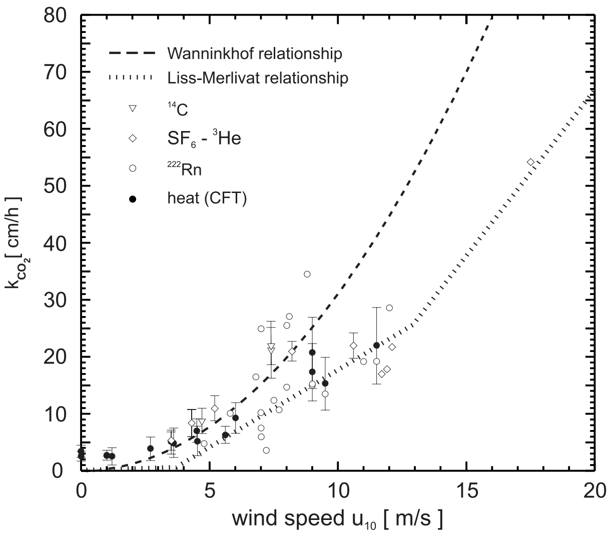


Figure 35.10: *Summary of gas exchange measurements in the ocean and transfer velocity/wind speed relations (white symbols), including heat exchange data (black dots).*

35.6 References

[1] Siegenthaler, U. and Sarmiento, J., (1993). Atmospheric carbon dioxide and the ocean. *Nature*, **365**.

[2] Keeling, R. and Shertz, S., (1992). Seasonal and interannual variations in atmospheric oxygen and implications for the global carbon cycle. *Nature*, **358**.

[3] Liss, P. and Duce, R., (1996). *The Sea Surface and Global Change*, first edition. Cambridge: Cambridge University Press.

[4] Jähne, B., Münnich, K., Bösinger, R., Dutzi, A., Huber, W., and Libner, P., (1987). On the parameters influencing air-water gas exchange. *Jour. Geophysical Research*, **92**(C2):1937–1949.

[5] Schimpf, U., Klinke, J., Haußecker, H., and Jähne, B., (1998). Novel instrumentation for synergetic studies of short wind waves, microturbulence and air sea gas transfer. *EOS*, **79**:107.

[6] Jähne, B., Libner, P., Fischer, R., Billen, T., and Plate, E., (1989). Investigating the transfer process across the free aqueous viscous boundary layer by the controlled flux method. *Tellus*, **41B**:177–195.

[7] Haußecker, H., Schimpf, U., and Jähne, B., (1998). Air sea gas transfer and the dynamics and patterns of the ocean surface micro turbulence. *Eos, Transactions*, **79**:37.

[8] Haußecker, H. and Jähne, B., (1995). In situ measurements of the air-sea gas transfer rate during the MBL/CoOP west coast experiment. In *Air-Water Gas Transfer. Selected Papers from the Third International Sym-*

posium on Air-Water Gas Transfer, B. Jähne and E. Monahan, eds., pp. 775–784. Hanau, Germany: Aeon.

[9] Jähne, B. and Haußecker, H., (1998). Air-water gas exchange. *Annu. Rev. Fluid Mech.*, **30**:443–468.

[10] BitFlow, (1995). *Raptor User Manual.* Woburn, MA: BitFlow Inc.

[11] ML2, (1995). *Multi-Lab/2 User Manual.* Heidelberg, Germany: SORCUS Computer GmbH.

[12] GSI, (1991). *XY Scan Head Series User Manual.* Watertown, MA: General Scanning Inc.

[13] Haußecker, H., (1996). *Messung und Simulation von kleinskaligen Austauschvorgängen an der Ozeanoberfläche.* Dissertation, Interdisciplinary Center for Scientific Computing, University of Heidelberg, Germany.

[14] Schimpf, U., (1996). *Fourieranalyse mikroskaliger Temperaturfluktuationen der Wasseroberfläche.* Master's thesis. University of Heidelberg, Germany: Interdisciplinary Center for Scientific Computing.

[15] AMBER, (1993). *Radiance 1 Smart Infrared Camera System.* Goleta, CA: Amber Engineering.

36 Thermography to Measure Water Relations of Plant Leaves

Bernd Kümmerlen[1,2], Stefan Dauwe[1,2],
Dominik Schmundt[1,2], and Ulrich Schurr[1]

[1]Botanisches Institut, Universität Heidelberg, Germany
[2]Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR)
Universität Heidelberg, Germany

36.1 Botanical background

Temperature relations in plants are very closely linked to their water relations. In contrast to, for example, mammals, plants are poikilothermic organisms, which means that their temperature is determined mainly by their environment because internal heat sources due to metabolism are of minor importance for the *energy budget*. There are very few exceptions from this rule (for example, in Arum species, which produce heat in their blossom to attract pollinating insects). However, this temperature increase can only be maintained for a few days and the entire organ "burns" its energy during this time.

Because plants are sessile organisms and, thus, cannot escape from unfavorable environments, they either have to face the temperature in

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

ISBN 0–12–379773-X/$30.00

their habitat or flee by growing only during periods when favorable conditions are present. Examples for the latter are annual plants that survive unfavorable conditions as seeds or biannual species that survive cold periods, for example, as bulbs or rhizomes [1].

Passive thermoregulation functions involve increasing the proportion of reflected radiation (e. g., by white hairs, salt crusts, etc.) or diminishing exposure by altering the inclination. The only short-term thermoregulation in plants is to alter *transpiration*. However, this is not a sensible possibility under severe heat stress, which is often accompanied by drought. As uncontrolled water loss to the relatively dry atmosphere will cause severe damage, the above-ground surfaces of plants are covered with a waxy layer (cuticle) that strongly prevents water evaporation from the plant corpus. *Transpiration* occurs through pores (*stomata*) formed by a complex of cells in the outermost cell layer of leaves. Water is driven out of the leaf by the strong difference in water concentration between the leaf-internal gas spaces and the surrounding atmosphere. The cells that form the stomata control transpirational water loss by opening or closing the aperture of the pores and hence altering the conductance of the leaf surface for water vapor [2].

Apart from the impact on leaf temperature, stomata integrate many important functions for the plant and are, thus, highly controlled by external and internal factors: CO_2 enters the leaf for photosynthetic assimilation via the same pores. Thus, closure of stomata to decrease water loss will simultaneously decrease the conductance for CO_2 through the leaf surface. Additionally, the transpiration stream provides the major route of nutrient transport from the root system to the shoot and, therefore, alterations of transpiration have consequences for the nutrient relations of the plant shoot. For a wide range of physiological functions in the plant, transpiration is very important and has been studied intensively. Nevertheless, the currently available techniques [3] have significant drawbacks (see Section 36.2).

A direct link between thermal properties of the plant and water relations is that the absorption of radiation in the infrared occurs mainly by water. With the exception of some particular states like seeds, the water content in plants is high. As can be seen in Fig. 36.1, a plant leaf has a heterogeneous internal structure. Different parts contain different amounts of water per area, affecting the thermal properties of the plant organs. Additionally, insulating structures like intercellular gas spaces are present, which may alter the heat transfer. These thermal properties of the leaf have not yet been studied intensively because appropriate techniques were missing.

Thermography matches several important requirements for botanical research. As a remote sensing technique it interferes as little as possible with the environment of the plant. Furthermore, it provides high temporal and spatial resolution. These aspects become highly relevant with the increasing awareness of the importance of dynamic processes and their distribution for the behavior of plants. Section 36.2 will give a survey of previously used methods to measure water relations of plant leaves.

Infrared imaging has been applied to plants previously by Hashimoto et al. [4], but only in a very qualitative sense. In order to connect the infrared images with physical parameters of the plant tissue, a thorough theoretical analysis of the thermal properties has to be done to analyze which parameters need to be measured to obtain meaningful results and to establish a sound physical basis for the interpretation of the infrared images. This theoretical analysis is done in Section 36.3. The actual thermography measurements are described in Section 36.4.

36.2 Previous measurement techniques

36.2.1 Transpiration measurements

The simplest, but nevertheless in many cases most suitable, technique to analyze transpiration is to weigh the plants. This low-tech analysis is simple to install and delivers relevant quantitative data, if certain pitfalls, like evaporation from the soil, are circumvented. However, the resulting data have no spatial resolution at all and it is impossible to apply this technique in field studies. Classical *gas exchange* measurements involve enclosing the organ of interest (e. g., a leaf) in a cuvette and analyzing the change of the composition of gas pumped through the cuvette. This technique has found wide application in botanical research, as it allows for study of gas exchange properties of individual parts of the plant including dynamic responses to environmental conditions. Differences in the water concentration of the gas transported through the cuvette are determined by infrared gas analyzers and multiplied by the gas flux rate to obtain the water flux out of the leaf [5]. The same approach is applied for analyzing assimilation rates of CO_2. Several variations of this principle are applied and available as commercial instruments. However, spatial resolution is poor and depends on the cuvette system.

High spatial resolution analysis of gas exchange of leaves can be obtained by *chlorophyll fluorescence* [6]. This approach is rather indirectly related to stomatal conductance. In principle, light energy absorbed by the photosynthetic pigments of the leaf (mainly chlorophyll) can be transferred into chemical energy by the photosynthetic process. Alternatively and in many cases more likely is the emittance of the absorbed energy as heat or as fluorescent light. The proportion of energy emitted via this pathway increases when fixation of CO_2 by the biochemical part of photosynthesis is inhibited. This is the case—besides other cases—when stomata are closed, as the leaf internal concentration of CO_2 declines and less substrate for photosynthetic CO_2 fixation is present. While this technique is able to map photosynthesis relatively accurately, the analysis of transpiration rates or, better, stomatal conductance is a rather indirect measure.

Stomatal conductance has also been determined by measuring leaf temperature [7]. This technique provided only very poor spatial and temporal resolution.

36.2.2 Water content and leaf thickness

Approximately 85–95 % of a plant's fresh weight is water. This parameter is widely used to judge the water status of plant tissue and is commonly determined after destructive sampling of plant pieces, analysis of the fresh weight and subsequently the dry weight of the tissue. As the determination involves destructive treatment, relatively little is known on the dynamics of this parameter, for example, during the diurnal variation of water status.

The leaf water content per unit gives additional information on the local leaf thickness. A specific number of cell layers make up the leaf thickness and, thus, the local *water content* contains additional information on the structure of the leaf. This structure is usually determined with destructive microscopical techniques which are laborious and do not allow sufficiently high temporal resolution to connect leaf thickness with dynamic variation of the leaf water status. Thus, a remote sensing technique measuring leaf thickness will deliver relevant information on the leaf structure.

36.3 Theoretical background

Leaf temperature is the net result of several energy fluxes into and out of the leaf. The magnitude of these fluxes is determined by a set of external and internal parameters. The only parameter that can be controlled by the plant itself in short-term is the aperture of the stomata, which controls the flux of gases such as CO_2 and water vapor through the leaf surface. As the heat of vaporization of water is very high (the energy needed to evaporate 1 mg of water is sufficient to cool down almost 600 mg of water by 1 K), the evaporation of water from internal cell walls causes an apparent flux of energy out of the leaf.

In the following part, the energy balance equation will be deduced to determine actual transpiration rates from infrared images of a leaf and a nontranspiring reference body. For a more thorough analysis see Kümmerlen [8].

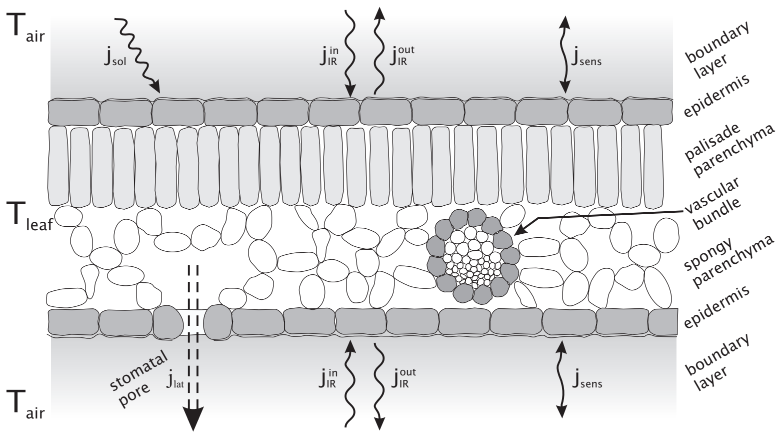


Figure 36.1: *Schematic cut through a leaf. It shows four distinct layers: the upper and lower epidermis which protect the leaf, the palisade parenchyma where most of the photosynthesis takes place and the spongy parenchyma where the gas exchange can take place. For an explanation of the energy fluxes see text.*

36.3.1 Leaf energy budget

As the first law of thermodynamics states, the total amount of energy in a closed system is conserved. It can only be converted between different forms. Therefore, a simple formulation of the energy balance is given by:

$$\text{energy storage} = \text{energy influx} + \text{energy efflux} \qquad (36.1)$$

In this chapter, influxes will be handled as positive and effluxes as negative terms in the equations. The energy-storage component consists of *photosynthesis*, other metabolical processes and *changes in the leaf temperature*. Photosynthesis and other metabolical processes can be neglected (they typically account for less than 1 % of the influxes) [2, p. 347].

Influx of energy results from *absorbed radiation* (from the entire spectrum, visible to thermal infrared). The main part of the energy *effluxes* is heat loss through water evaporation. In addition, the leaf emits *infrared radiation* according to the *Stefan-Boltzmann law* Eq. (36.6). Heat conduction and convection can either be an influx or an efflux, depending on the temperature difference between the leaf and the surroundings. Taking into account all these terms, Eq. (36.1) becomes

$$j_{net} = j_{sol} + j_{IR} + j_{sens} + j_{lat} \qquad (36.2)$$

The different terms in this energy balance equation are depicted in Fig. 36.1 and will be explained in the following paragraphs. For easier comparison, all energy fluxes are expressed as flux *densities* (in Wm^{-2}).

Temperature changes. The temperature change of an object when a certain amount of heat is supplied can be determined with a simple relationship:

$$\mathrm{d}T = \frac{1}{C}\,\mathrm{d}Q \tag{36.3}$$

where C is the *heat capacity* of the object and dQ is the heat amount (energy). The net energy flux density, j_{net}, of a given energy flux Φ through an area A is:

$$j_{\mathrm{net}} := \frac{\Phi}{A} = \frac{1}{A}\frac{\mathrm{d}Q}{\mathrm{d}t} = \frac{C}{A}\frac{\mathrm{d}T}{\mathrm{d}t} \tag{36.4}$$

C/A is the heat capacity per unit area of the object. Equation (36.4) directly relates temporal temperature changes to the net energy flux density j_{net}.

Solar irradiation. The term *solar irradiation* denominates the energy influx by radiation in the visible part of the spectrum. For a plant under field conditions this would normally be direct sunlight. These energy flux densities can vary over a big range, from 1200 Wm^{-2} (in full sunlight) to zero (during the night). In a climate chamber situation, which is often used in botanical studies to control climatic conditions, artificial light is supplied (which amounts to 50–500 Wm^{-2}).

Not all of this energy is absorbed by the leaf, which is taken into account by the *absorptivity* α_{sol}. The influx can be written as:

$$j_{\mathrm{sol}} = \alpha_{\mathrm{sol}} j_{\mathrm{in}} \tag{36.5}$$

with the incident radiation flux density j_{in}.

Infrared radiation. Every object emits thermal radiation (see Volume 1, Section 2.5). For a *blackbody* the total emitted energy flux density j_{bb} can be calculated using the *Stefan-Boltzmann law* (see Equation (2.41)):

$$j_{bb} = \sigma T_{bb}^4 \tag{36.6}$$

Under field conditions infrared radiation from the sky has to be taken into account. This radiation originates mainly from H_2O and CO_2 molecules in the atmosphere. The effective temperature of the sky (i. e., the temperature for which $\sigma T_{\mathrm{sky}}^4$ equals the radiant energy from the sky) can vary between 220 K (around -50°C for clear sky) and 280 K (around +5 °C, cloudy) [2, p. 352].

The surroundings can be considered as an object with the effective temperature T_{surr}. For simplification we assume that only the upper leaf surface receives IR radiation from the sky and only the lower leaf surface is subjected to T_{surr}. Therefore, the absorbed infrared radiation would be $j_{\mathrm{IR}}^{\mathrm{field}} = \alpha_{\mathrm{IR}}\sigma(T_{\mathrm{surr}}^4 + T_{\mathrm{sky}}^4)$. In the climate chamber the part of the sky is given by the ceiling, which ideally has the same temperature as the surroundings. This leads to

$$j_{\mathrm{IR}}^{\mathrm{in}} = 2\alpha_{\mathrm{IR}}\sigma T_{\mathrm{surr}}^4 \tag{36.7}$$

σ is the Stefan-Boltzmann constant, α_{IR} the leaf absorptivity in the thermal infrared.

The leaf also emits infrared radiation according to its temperature. Both leaf surfaces emit into one hemisphere, the total emitted energy flux density when taking into account *Kirchhoff's radiation law* (which states that the *absorptivity* α_{IR} and *emissivity* ϵ_{IR} for a thermal radiator are the same at each wavelength) is

$$j_{\mathrm{IR}}^{\mathrm{out}} = -2\alpha_{\mathrm{IR}}\sigma T_{\mathrm{leaf}}^4 \tag{36.8}$$

Combining the two infrared radiation terms of Eqs. (36.7) and (36.8) gives the net IR radiation flux density as

$$j_{\mathrm{IR}} = 2\alpha_{\mathrm{IR}}\sigma(T_{\mathrm{surr}}^4 - T_{\mathrm{leaf}}^4) \tag{36.9}$$

Heat conduction and convection. Two mechanisms can transfer heat between adjacent media: conduction and convection, often referred to as *sensible heat transport*. At the solid-gas interface, a *boundary layer* exists, over which heat is transferred only by conduction. The thickness of the boundary layer (and, therefore, the *transfer velocity* of heat across it) is strongly dependent on the wind speed over the leaf. Beyond the boundary layer, heat is transported away from the leaf by turbulent convection. This is generally a very fast process, which makes the conductive part the main resistance for sensible heat transport.

Fick's first law describes the diffusive transport of substances in the presence of a density gradient (here 1-D)

$$j_{\mathrm{diff}} = -D\frac{\mathrm{d}}{\mathrm{d}x}c$$

with c: concentration; D: diffusion coefficient; j_{diff}: flux density.

When this law is applied to heat transfer of the leaf surface, “concentration” becomes energy density and flux density becomes energy flux density. The energy density stored in a medium can be calculated as $c = \rho c_p T$, which leads to

$$j_{\mathrm{diff}}^{\mathrm{heat}} = -D\rho\, c_p\, \frac{\mathrm{d}T}{\mathrm{d}x} \tag{36.10}$$

Here ρ is the density of the medium and c_p is its specific heat capacity. This relationship is also called Fourier's heat transfer law.

The *transfer velocity* k_heat can be defined as the ratio of the diffusion coefficient D to the diffusion length Δx:

$$k_\text{heat} = -\frac{D}{\Delta x} \tag{36.11}$$

The heat flux density over the leaf-air boundary layer can then be expressed as follows, with the factor of 2 indicating the flux over both leaf surfaces:

$$j_\text{sens} = -2k_\text{heat}\,\rho\,c_p\,(T_\text{leaf} - T_\text{air}) \tag{36.12}$$

Now ρ and c_p are the density and specific heat capacity of air. The direction of this energy flux depends obviously on the temperature of the leaf and the surrounding air.

Transpiration cooling. The phase transition of water from liquid to vapor consumes a considerable amount of energy called *latent heat*. Again, using *Fick's first law*, the mass flux density j_{wv} of water vapor can be calculated for given concentrations. Similar to Eq. (36.11) the transfer velocity for water vapor is given by $k_{wv} = -\frac{D_{wv}}{\Delta x}$:

$$j_{wv} = -D_{wv}\,\frac{\Delta c_{wv}}{\Delta x} = k_{wv}\,\Delta c_{wv}$$

Δc_{wv} is the difference between water vapor concentrations inside the leaf and in the surrounding turbulent air. The concentrations can be expressed as relative humidities (fractions of the water vapor saturation concentration at the given temperature):

$$j_{wv} = k_{wv}\,c_{wv}\,(h_\text{air} - h_\text{leaf})$$

The heat loss is calculated as the product of this flux density and the heat of evaporation of water (with a negative sign because efflux of water vapor from the leaf causes energy loss from the system)

$$j_\text{lat} = -\lambda\,j_{wv} \tag{36.13}$$

Traditionally, water vapor fluxes are expressed in molar units, which are achieved if j_{wv} is multiplied with the molecular weight of water MW_{H_2O}:

$$j_\text{lat} = -\lambda\,MW_{H_2O}\,j_{wv}^\text{mol} \tag{36.14}$$

36.3.2 Calculation of transpiration rates

Equation (36.2) can now be expressed by the individual contributing terms:

$$\begin{aligned}\left(\frac{C}{A}\right)_\text{leaf}\frac{dT_\text{leaf}}{dt} &= \alpha_\text{sol}^\text{leaf}\,j_\text{in} + 2\alpha_\text{IR}^\text{leaf}\,\sigma\,(T_\text{surr}^4 - T_\text{leaf}^4)\\ &\quad + 2k_\text{heat}^\text{leaf}\,\rho\,c_p\,(T_\text{air} - T_\text{leaf}) + k_{wv}\,\lambda\,c_{wv}\,(h_\text{leaf} - h_\text{air})\end{aligned} \tag{36.15}$$

For determination of *transpiration rates* this equation can be solved for j_{wv}^{mol} according to Eq. (36.14). For reasons that will be clarified in the next paragraphs, indices were added to associate certain properties with the leaf.

This equation contains some terms that are variable and hard to measure. Especially the incident solar flux j_{in} and the absolute temperature of the surroundings T_{surr} can not be measured with sufficient accuracy. The leaf temperature T_{leaf} can be determined directly from infrared images acquired with an infrared camera (see Section 36.4). Although the thermal resolution of the infrared camera is very good, absolute temperatures are just as accurate as the camera's calibration, which is normally about 0.1 K. To get the most accurate measurements, a relationship between a temperature difference (which can be measured with an accuracy of 0.025 K with the camera) and the transpiration rate needs to be established.

One way to achieve this is to consider a nontranspiring object, which is subject to the same energy fluxes as the leaf. For this reference object an equation similar to Eq. (36.15) is obtained:

$$\left(\frac{C}{A}\right)_{\text{ref}} \frac{\mathrm{d}T_{\text{ref}}}{\mathrm{d}t} = \alpha_{\text{sol}}^{\text{ref}} j_{\text{in}} + 2\alpha_{\text{IR}}^{\text{ref}} \sigma (T_{\text{surr}}^4 - T_{\text{ref}}^4) + 2k_{\text{heat}}^{\text{ref}} \rho\, c_p\, (T_{\text{air}} - T_{\text{ref}}) \tag{36.16}$$

Compared to Eq. (36.15), the latent heat flux is missing since no transpiration occurs from the object.

If both the leaf and the object are in the same environment, T_{air} and T_{surr} can be considered equal in both equations and, therefore, one of these temperatures can be eliminated to obtain one single equation. If this is done for T_{surr}^4, we get

$$\begin{aligned}\left(\frac{C}{A}\right)_{\text{leaf}} \frac{\mathrm{d}T_{\text{leaf}}}{\mathrm{d}t} - \frac{\alpha_{\text{IR}}^{\text{leaf}}}{\alpha_{\text{IR}}^{\text{ref}}} \left(\frac{C}{A}\right)_{\text{ms}} \frac{\mathrm{d}T_{\text{ref}}}{\mathrm{d}t} &= \left(\alpha_{\text{sol}}^{\text{leaf}} - \frac{\alpha_{\text{IR}}^{\text{leaf}}}{\alpha_{\text{IR}}^{\text{ref}}} \alpha_{\text{sol}}^{\text{ref}}\right) j_{\text{in}} \\ &+ 2\alpha_{\text{IR}}^{\text{leaf}} \sigma \left(T_{\text{ref}}^4 - T_{\text{leaf}}^4\right) + 2\rho c_p \left(k_{\text{heat}}^{\text{leaf}} - \frac{\alpha_{\text{IR}}^{\text{leaf}}}{\alpha_{\text{IR}}^{\text{ref}}} k_{\text{heat}}^{\text{ref}}\right) T_{\text{air}} \\ &+ 2\rho c_p \left(\frac{\alpha_{\text{IR}}^{\text{leaf}}}{\alpha_{\text{IR}}^{\text{ref}}} k_{\text{heat}}^{\text{ref}} T_{\text{ref}} - k_{\text{heat}}^{\text{leaf}} T_{\text{leaf}}\right) + j_{\text{lat}}\end{aligned} \tag{36.17}$$

The temperature difference between the reference body and the leaf is $\Delta T = T_{\text{ref}} - T_{\text{leaf}}$. For small ΔT (< 5 K), the following approximation applies:

$$T_{\text{ref}}^4 - T_{\text{leaf}}^4 \approx 4T_{\text{ref}}^3 \Delta T \tag{36.18}$$

Combining this with Eq. (36.17) and Eq. (36.14), a relationship for the transpiration rate (water vapor flux) is obtained:

$$\begin{aligned} j_{wv}^{\mathrm{mol}} &= \frac{1}{\lambda MW_{\mathrm{H_2O}}} \Bigg\{ \left(8\alpha_{\mathrm{IR}}^{\mathrm{leaf}} \sigma T_{\mathrm{ref}}^3 + 2\rho c_p k_{\mathrm{heat}}^{\mathrm{leaf}} \right) \Delta T \\ &+ \left(\frac{C}{A} \right)_{\mathrm{leaf}} \frac{\mathrm{d}T_{\mathrm{leaf}}}{\mathrm{d}t} - \frac{\alpha_{\mathrm{IR}}^{\mathrm{leaf}}}{\alpha_{\mathrm{IR}}^{\mathrm{ref}}} \left(\frac{C}{A} \right)_{\mathrm{ms}} \frac{\mathrm{d}T_{\mathrm{ref}}}{\mathrm{d}t} \\ &+ 2\rho c_p \left(k_{\mathrm{heat}}^{\mathrm{leaf}} - \frac{\alpha_{\mathrm{IR}}^{\mathrm{leaf}}}{\alpha_{\mathrm{IR}}^{\mathrm{ref}}} k_{\mathrm{heat}}^{\mathrm{ref}} \right) (T_{\mathrm{air}} - T_{\mathrm{ref}}) \\ &+ \left(\alpha_{\mathrm{sol}}^{\mathrm{leaf}} - \frac{\alpha_{\mathrm{IR}}^{\mathrm{leaf}}}{\alpha_{\mathrm{IR}}^{\mathrm{ref}}} \alpha_{\mathrm{sol}}^{\mathrm{ref}} \right) j_{\mathrm{in}} \Bigg\} \end{aligned} \tag{36.19}$$

This is clearly a linear relationship between ΔT and the latent heat flux. Two parameters in Eq. (36.19) cannot be easily obtained: the heat transfer velocity over the boundary layer $k_{\mathrm{heat}}^{\mathrm{leaf}}$, and the heat capacity per unit leaf area $(C/A)_{\mathrm{leaf}}$. These properties have to be determined experimentally. Most of the other terms can be neglected if the reference body is chosen so that its optical properties are very similar to those of the leaf (i. e., $\alpha_{\mathrm{IR}}^{\mathrm{ref}} = \alpha_{\mathrm{IR}}^{\mathrm{leaf}}$ and $\alpha_{\mathrm{sol}}^{\mathrm{ref}} = \alpha_{\mathrm{sol}}^{\mathrm{leaf}}$).

36.4 Measurements

In the context of the theoretical considerations in Section 36.3, the measurements conducted with the Amber Radiance 1 infrared camera had two aims:

The linearity of the relationship between ΔT and transpiration rate had to be proven experimentally. This was done by a series of measurements in which leaf temperature and transpiration rates were measured simultaneously. As the plant is just observed and no external energy fluxes are applied, this technique is termed *passive thermography*.

In order to determine the heat capacity per unit area $(C/A)_{\mathrm{leaf}}$, the energy equilibrium of the leaf had to be actively perturbed by application of additional infrared energy fluxes (therefore the term *active thermography*). The observed local changes in leaf temperature should allow for heat capacity on the leaf to be mapped.

36.4.1 Passive thermography

In Section 36.3.2, a linear relationship between transpiration rate and the temperature difference between a nontranspiring reference object and the leaf has been deduced theoretically. To confirm this relationship, it is necessary to measure the *transpiration rates* of the leaf with a standard method along with the temperatures of the leaf and the reference body.

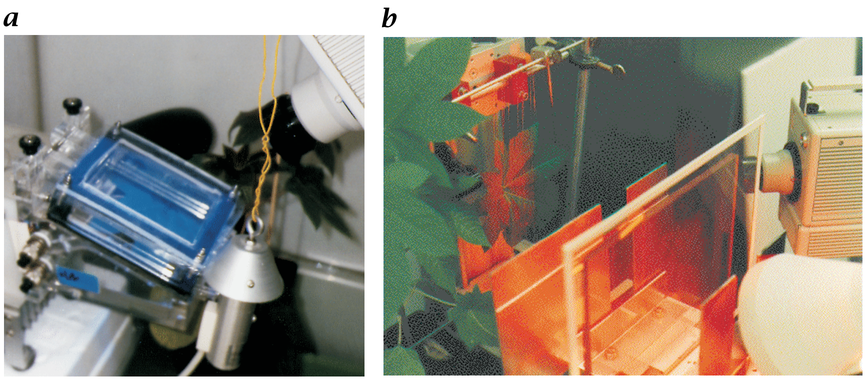


Figure 36.2: ***a*** *Passive thermography setup, the infrared camera is in the upper right corner, the plant in the back has one leaf fixed inside the cuvette;* ***b*** *active thermography setup, the infrared radiator illuminates the leaf through a set of screen and filters.*

Experimental setup. For the measurement of the transpiration rate, the tip of the central lobe of the leaf of *Ricinus communis* plants was clamped between the two halves of a standard *gas exchange* cuvette made from acrylic glass (see Fig. 36.2a). To reduce disturbing environmental influences, this system was placed in a 1.5 m^3 thermally insulated chamber with humidity and temperature control and an artificial light source.

The gas exchange system provided control over the relevant external parameters including humidity, temperature and wind speed inside the cuvette. The leaf could also be supplied with specific gas mixtures through a gas mixing unit, thus allowing control of the CO_2 concentration around the leaf. The cuvette measured all the relevant properties such as internal and external air temperatures, air humidity, light intensity, the concentrations of water vapor and CO_2 and the concentration difference of these two gases between the input and output gas paths.

In addition to the leaf, a small brass plate (1 cm^2) was placed in the cuvette as temperature reference blackbody. The plate was coated with Tetenal Photolack, a black varnish with an infrared emissivity of 97 %.

The entire setup was imaged with the Amber Radiance 1 infrared camera. Image sequences consisting of 256 images, were acquired in 10-s intervals. This yields a sequence length of approximately 43 min.

Inducing stomatal responses. In order to induce different transpiration rates, blue light pulses were used [9]. As stomata continue to open in response to the blue light pulse, even after the pulse has been applied, the responses of leaf temperature to direct energy input during the light pulse and changes in leaf temperature due to opening of the stomata could be separated.

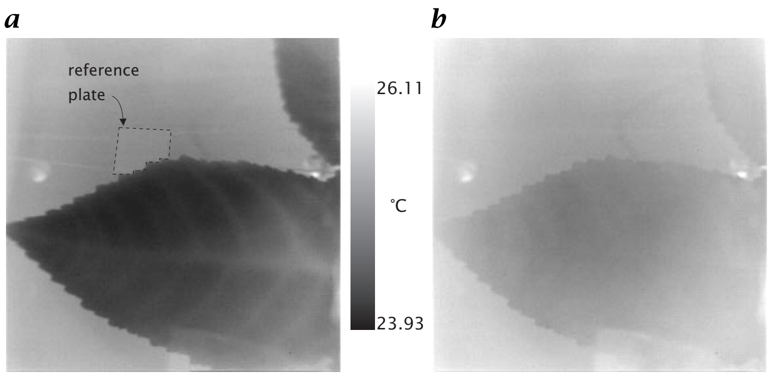


Figure 36.3: *Infrared images of the light pulse sequence:* ***a*** *at 20 min: transpiration is high, therefore, the leaf is cool. The reference plate is indicated above the leaf;* ***b*** *at 40 min: stomata are nearly closed, leaf temperature is very similar to the reference plate temperature.*

The light pulse was given by a slide projector suspended above the cuvette. A filter restricted the incident light to a wavelength of 350–460 nm (blue). This was necessary to reduce the amount of heat put into the leaf by the projector without losing control on stomatal aperture.

After acclimatization of the plant to the cuvette, the image acquisition was started. After 5 min, the blue light was switched on for about 5 min. For the rest of the time the plant received very little light from its surroundings.

Results. Two images of the acquired image sequence are presented in Fig. 36.3. Leaf temperature was always below the temperature of the reference plate. This is expected because of the latent energy flux out of the leaf. Furthermore, the vein area is significantly warmer because the transpiration is supposed to be smaller there due to lower stomatal density. After the application of the light pulse at 5 min all temperatures rose due to the additional energy flux from the illumination.

For each image in the sequence, $\Delta T = T_{\mathrm{ref}} - T_{\mathrm{leaf}}$ was calculated for each pixel by subtracting the entire image from T_{ref}, which resulted in an image sequence showing ΔT over time. Plotting the transpiration rate against ΔT yields a graphical representation of Eq. (36.19). This plot (Fig. 36.4) depicts two features of this method of inducing stomatal opening: The quick change in illumination at 5 and 10 min changes the energy balance of the leaf considerably. The incident flux j_{in} increases, which leads to an increase in the temperatures. Due to the fast temperature changes, the differences in heat capacity between the leaf and the reference plate gain importance and disturb the linear relationship (circles in Fig. 36.4). The differences in the absorptivities of leaf and

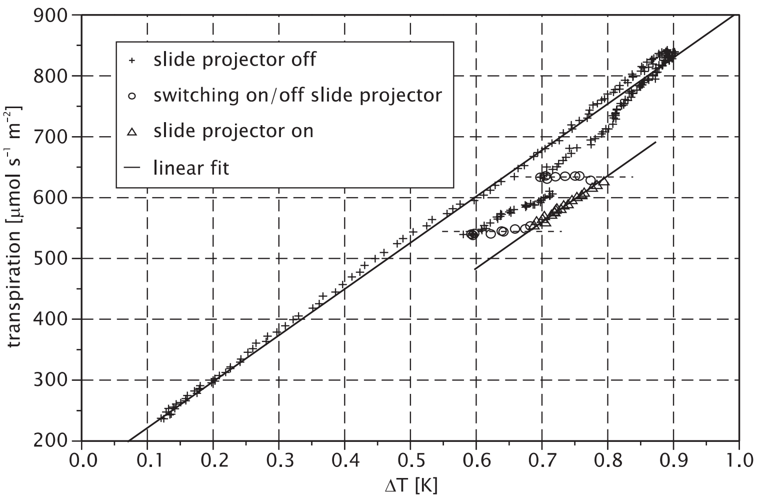


Figure 36.4: *Transpiration rate plotted against temperature difference. For the unperturbed phases of the experiment, the linear relationship Eq. (36.19) is obvious. During fast temperature changes (circles) the relationship is not valid due to differences in heat capacity. When the slide projector is on (triangles), the temperature difference is higher due to the higher absorptivity of the reference object in the visible part of the spectrum.*

reference lead to a bigger temperature difference during the light pulse, nonetheless the slope of the linear relation stays the same (triangles in Fig. 36.4).

Omitting the parts of the measurement where fast changes in temperature or radiation occurred, the transfer velocity for heat across the leaf-air boundary layer could be determined from the slope of the linear fit in Fig. 36.4. This slope is $(752.2 \pm 5.0)\,\mu\mathrm{mol\,s^{-1}\,m^{-2}\,K^{-1}}$. Together with Eq. (36.19), a value of

$$k_{\mathrm{heat}}^{\mathrm{leaf}} = 8.84\,\frac{\mathrm{mm}}{\mathrm{s}} \tag{36.20}$$

was established for an estimated wind velocity of $1.4\,\mathrm{ms^{-1}}$ inside the cuvette. This is compatible with results quoted in Linacre [10]. Due to the design of the cuvette, wind velocities are not known accurately and can not be measured.

Future developments. To establish the linear relationship Eq. (36.19) experimentally, it is necessary to change stomatal aperture without changing the energy balance of the leaf. For radiation changes this was clearly not the case. However, stomata also respond to CO_2-concentration of the surrounding air. If this concentration is lowered, stomates open to compensate for the decrease in internal CO_2-concentration.

This change has no effect on any of the energy fluxes except the latent heat flux (transpiration), which makes it the ideal means of inducing stomatal responses. First experiments have been made using this technique with matching results.

It is obvious that fast changes of temperatures or irradiation can not be avoided in a field situation with natural light. Therefore, it is necessary to determine $(C/A)_{\text{leaf}}$, which can vary considerably over the leaf due to differences in internal leaf structure. A means to measure the heat capacities with spatial resolution by application of active thermography is described in what follows.

36.4.2 Active thermography

For the active thermography to measure heat capacities of leaves, the response of the leaf temperature to an imposed energy flux was observed. The leaf energy budget (Eq. (36.15)) gives the key to the determination of the heat capacity.

Under the assumption $T_{\text{surr}} = T_{\text{air}}$, this equation can be written using $\delta T = T_{\text{air}} - T_{\text{leaf}}(t)$

$$\begin{aligned} -\frac{\mathrm{d}(\delta T(t))}{\mathrm{d}t} &\approx \left(\frac{A}{C}\right)_{\text{leaf}} \Big[\alpha_{\text{sol}}^{\text{leaf}} j_{\text{in}} + j_{\text{lat}} + j_{\text{act}}(t) \\ &+ \left(8\alpha_{\text{IR}}^{\text{leaf}} \sigma T_{\text{air}}^3 + 2\rho c_p k_{\text{heat}}^{\text{leaf}}\right) \delta T(t)\Big] \end{aligned} \tag{36.21}$$

An additional heat flux density $j_{\text{act}}(t)$ is introduced as an actively controllable heat source. Again, as in Eq. (36.18), the approximation $T_{\text{air}}^4 - T_{\text{leaf}}^4(t) \approx 4T_{\text{air}}^3 \delta T(t)$ was used, which is valid for small δT.

Using the abbreviations

$$\tau := \left(\frac{C}{A}\right)_{\text{leaf}} \frac{1}{\left(8\alpha_{\text{IR}}^{\text{leaf}} \sigma T_{\text{air}}^3 + 2\rho c_p k_{\text{heat}}^{\text{leaf}}\right)} \tag{36.22}$$

$$b := \left(\frac{A}{C}\right)_{\text{leaf}} \left(\alpha_{\text{sol}}^{\text{leaf}} j_{\text{in}} + j_{\text{lat}}\right) \tag{36.23}$$

$$z(t) := \left(\frac{A}{C}\right)_{\text{leaf}} j_{\text{act}}(t) \tag{36.24}$$

Equation (36.21) becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}\delta T(t) = -\frac{1}{\tau}\delta T(t) - b - z(t) \tag{36.25}$$

When $z(t)$ is chosen appropriately this equation can easily be solved. Two choices of $z(t)$ will be discussed here: a step change in $j_{\text{act}}(t)$ and a periodic variation.

Setup. The additional energy flux $j_{\text{act}}(t)$ was applied with a medical infrared radiator (Philips 150W R95E). To avoid direct reflexes by the lamp on the leaf, a set of screens and filters was placed in the optical path of the radiator (see Fig. 36.2b). The filters limited the radiation to which the leaf was actually exposed to wavelengths outside the sensitivity range of the camera, which is 3–5 μm. The radiation reaching the camera, therefore, is the true thermal infrared radiation emitted by the leaf. The radiator could be switched on and off from within the image processing software, allowing easy synchronization of image acquisition with radiation changes. Infrared image sequences of different length were acquired with frequencies of 0.1–10 Hz. For easier evaluation, all sequences consisted of 256 images.

Constant flux method. The infrared radiator is switched on at $t = 0$. This results in a constant additional energy flux density j_{act} for $t > 0$. The leaf temperature reaches a new equilibrium with time.

The solution for Eq. (36.25) in this case is

$$\delta T(t) = \Delta T\, e^{-\frac{t}{\tau}} + \delta T^{\infty} \tag{36.26}$$

where ΔT is the temperature increase of the leaf due to the additional flux and δT^{∞} the temperature difference between air and leaf in the new steady state. It can be shown that $\Delta T = z\tau$ [11]. For absolute temperatures, using $\delta T = T_{\text{air}} - T_{\text{leaf}}(t)$, Eq. (36.26) becomes

$$T_{\text{leaf}}(t) = T^{\infty}_{\text{leaf}} - \Delta T\, e^{-\frac{t}{\tau}} \tag{36.27}$$

Acquisition of image sequences started at $t = 0$ in synchronicity with the additional energy flux. For evaluation, the exponential function Eq. (36.27) was fitted to the temporal development of each pixel's temperature using the *Levenberg-Marquardt algorithm* for nonlinear least-squares fits [12]. This resulted in a set of parameter images containing the fit parameters T^{∞}_{leaf}, ΔT and τ (see Fig. 36.5).

An expression for the heat capacity is obtained from Eq. (36.22):

$$\left(\frac{C}{A}\right)_{\text{leaf}} = \tau\left(8\alpha^{\text{leaf}}_{\text{IR}}\sigma T^{3}_{\text{air}} + 2\rho c_p k^{\text{leaf}}_{\text{heat}}\right) \tag{36.28}$$

The heat capacity can be determined if the transfer velocity $k^{\text{leaf}}_{\text{heat}}$ is known. The quantities used to calculate the heat capacity have the following values: leaf absorptivity: $\alpha^{\text{leaf}}_{\text{IR}} = 0.975$; air temperature: $T_{\text{air}} = 298\,\text{K}$; heat transfer velocity: $k^{\text{leaf}}_{\text{heat}} = 8.84\,\text{mm}\,\text{s}^{-1}$; density of air: $\rho = 1.2\,\text{kg}\,\text{m}^{-3}$; specific heat capacity of air: $c_p = 1007\,\text{J}\,\text{kg}^{-1}\,\text{K}^{-1}$; time constant: $\tau_{ic} = 25\,\text{s}$, which leads to the following heat capacity for an interveinal area:

$$\left(\frac{C}{A}\right)^{\text{iv}}_{\text{leaf}} = 826\,\frac{\text{J}}{\text{Km}^{3}} \tag{36.29}$$

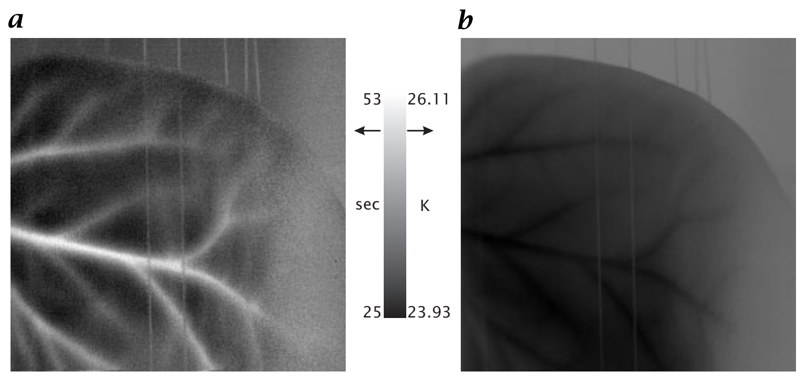


Figure 36.5: *Parameter images for constant heating:* ***a*** *time constant τ;* ***b*** *temperature difference ΔT.*

This would correspond to a water layer of

$$d_{iv} = \frac{\left(\frac{C}{A}\right)^{\text{iv}}_{\text{leaf}}}{C_p^{\text{water}}} = 197.8\,\mu\text{m}$$

($C_p^{\text{water}} = 4.165 \times 10^6\,\text{J}\,\text{K}\text{m}^{-3}$ is the specific heat capacity of water.) This calculated water layer thickness is in good agreement with microscopy data from the same species.

Periodic flux method. A varying energy flux density $j_{\text{act}}(t)$ is applied by switching the radiator periodically on and off. Therefore, $z(t)$ can not be easily omitted from Eq. (36.25). The *Fourier transform* theorem states that

$$\frac{\partial f(t)}{\partial t} \circ\!\!-\!\!\bullet \, \text{i}\omega \hat{f}(k) \tag{36.30}$$

that is, a time derivative of a function in the time domain becomes a multiplication of the Fourier transform of this function with iω in the frequency domain. Applied to Eq. (36.25) this yields

$$\text{i}\omega\widehat{\delta T}(\omega) = -\frac{1}{\tau}\widehat{\delta T}(\omega) - \hat{b} - \hat{z}(\omega) \tag{36.31}$$

Assuming constant energy flux densities j_{in} and j_{lat} and constant air temperature T_{air}, this can be rewritten for $\hat{T}_{\text{leaf}}$ as follows:

$$\hat{T}_{\text{leaf}}(\omega) = \frac{\hat{z}(\omega)}{\frac{1}{\tau} + \text{i}\omega} \tag{36.32}$$

The phase and amplitude of this complex quantity are:

$$\varphi(\omega) = -\arctan(\omega\tau) \quad \text{and} \quad \left|\hat{T}_{\text{leaf}}(\omega)\right| = \sqrt{\frac{\left|\hat{z}(\omega)\right|^2}{\frac{1}{\tau^2} + \omega^2}} \tag{36.33}$$

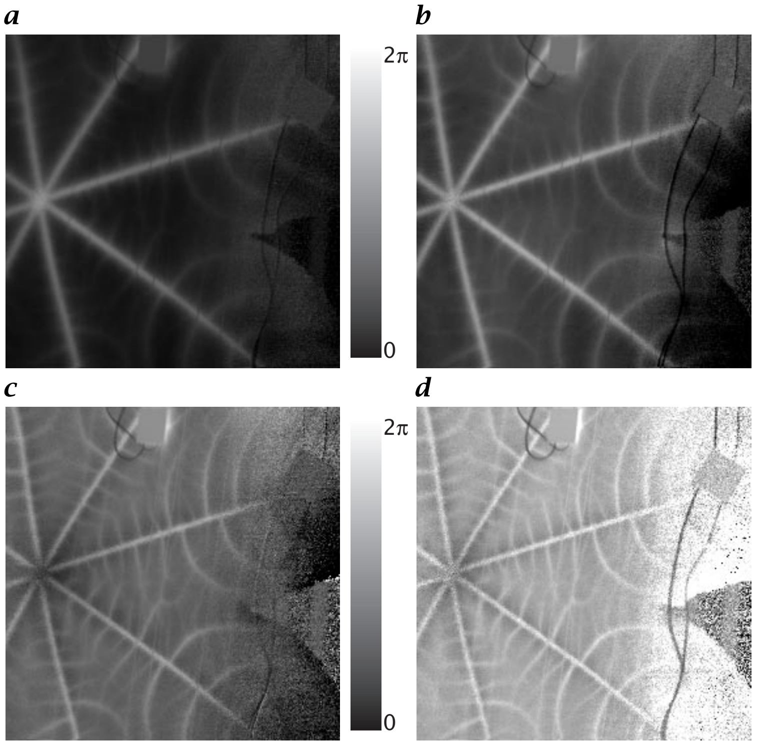


Figure 36.6: Parameter images depicting the phase Eq. (36.33) for different input radiation frequencies ω: ***a*** *$\omega = 2\pi\,0.0039$ rad/s;* ***b*** *$\omega = 2\pi\,0.0078$ rad/s;* ***c*** *$\omega = 2\pi\,0.0156$ rad/s;* ***d*** *$\omega = 2\pi\,0.0313$ rad/s; (see also Plate 12).*

For $\omega \ll 1/\tau$ (i. e., the period of the input signal is much longer than the time constant of the leaf), the phase becomes

$$\varphi(\omega) \approx -\omega\tau \tag{36.34}$$

In principle, a measurement of the phase $\varphi(\omega)$ yields a value for τ and, therefore, using Eq. (36.28), a value for $(C/A)_{\text{leaf}}$.

The varying energy flux had the form of a rectangular function (the radiator was periodically switched on and off). As the Fourier spectrum of a rectangular function consists of all the odd harmonics, this can be considered as an application of several frequencies at once (with decreasing amplitude).

Experiments were made with different input frequencies from 0.0039 to 0.0313 Hz, which corresponds to period lengths of 256 s to 32 s, respectively. For the base frequencies, the digital Fourier transform (see Section 3.3) was performed in the time direction. This resulted in parameter images containing the amplitude and the phase shift of the temperature signal of the leaf. Those two parameters can be visualized in a single color image: The amplitude is indicated by the brightness of the pixel, the color encodes phase shift.

Figure 36.6 shows the phase parameter images. For each frequency, the phase shift of the veins is bigger than the phase shift of the interveinal fields. Together with Eq. (36.34) and Eq. (36.28) this shows the bigger heat capacity of the veins.

36.5 Conclusion and further developments

On the basis of infrared thermography, two remote sensing techniques with a high spatial and temporal resolution for *transpiration* and *water content* have been developed. The principle of measurement was derived from the basic physical laws in Section 36.3. The measurements presented in Section 36.4 have confirmed these relations.

To study dynamic processes, it is necessary to combine the two techniques. This has already been realized in a new setup where passive and active thermography are integrated in a new gas exchange cuvette system [8]. Further development of this combination will make the remote sensing of the principal parameters of leaf gas exchange, water relations, and structural parameters feasible.

In passive thermography, improvements can be obtained by choosing an alternative method to induce stomatal opening, for example, a variation of CO_2-concentration around the leaf. This will vary the flux of latent heat (transpiration) without perturbation of the other energy fluxes.

Active thermography will benefit from the usage of a different heat source. This source is based on a *blackbody* and evades problems with the leaf absorptivity that arise from the utilization of a heat source with a high contribution in the visible and near infrared. Furthermore, it allows application of average free periodical signals, thus removing the heat stress on the plant.

With further theoretical and technical improvements, IR-imaging technology is a method with high potential benefit for plant sciences, especially for the analysis of dynamic processes in water relations and development.

Acknowledgments

The authors thank the German Science Foundation (DFG) for financial support of this research within the research unit "Image Sequence Analysis to Investigate Dynamic Processes."

36.6 References

[1] Larcher, W., (1994). *Ökophysiologie der Pflanzen*, 5. edition. Stuttgart: Ulmer Verlag.

[2] Nobel, P. S., (1991). *Physicochemical and Environmental Plant Physiology*. Boston: Academic Press.

[3] von Willert, D. J., Matyssek, R., and Heppich, W., (1995). *Experimentelle Pflanzenökologie*. Stuttgart: Georg Thieme Verlag.

[4] Hashimoto, Y., Ino, T., Kramer, P. J., Naylor, A. W., and Strain, B. R., (1984). Dynamic analysis of water stress of sunflower leaves by means of a thermal image processing system. *Plant Physiology*, **76**:266–269.

[5] von Caemmerer, S. and Farquhar, G., (1981). Some relationships between the biochemistry of photosynthesis and gas exchange of leaves. *Planta*, **153**:376–387.

[6] Siebke, K. and Weis, E., (1995). Assimilation images of leaves from Glechoma hedereceae. Analysis of non-synchronous stomata related oscillations. *Planta*, **196**:148–165.

[7] Impens, I. I., (1966). Leaf wetness, diffusion resistances and transpiration rates of bean leaves (Phaseolus Vulgaris L.) through comparison of “wet” and “dry” leaf temperatures. *Oeco. Planta.*, **I**:327–334.

[8] Kümmerlen, B., (1998). *Infrarot-Thermographie zum Studium physiologischer Parameter von Pflanzenblättern*. Diploma thesis, University of Heidelberg.

[9] Sharkey, T. D. and Raschke, K., (1981). Separation and measurement of direct and indirect effects of light on stomata. *Plant Physiology*, **68**:33–40.

[10] Linacre, E. T., (1967). Further studies of the heat transfer from a leaf. *Plant Physiology*, **42**:651–658.

[11] Dauwe, S., (1997). *Infrarotuntersuchungen zur Bestimmung des Wasser- und Wärmehaushalts eines Blattes*. Diploma thesis, University of Heidelberg.

[12] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., (1992). *Numerical Recipes in C: The Art of Scientific Computing*. New York: Cambridge University Press.

37 Retrieval of Atmospheric Trace Gas Concentrations

Carsten Leue, Mark Wenig, and Ulrich Platt

Institut für Umweltphysik, Universität Heidelberg, Germany

37.1 Introduction

Human activities have caused significant changes in the chemical composition of the earth's atmosphere in the past hundred years. Space borne measurement devices and techniques have become most important to monitor such changes.

Their history goes back more than 25 yr to the launch of the Backscatter Ultraviolet (BUV) instrument aboard the U.S. National Aeronautics and Space Administration's (NASA) Nimbus 4 satellite. Measurements have been ongoing since the late 1970s, largely through instruments

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

ISBN 0–12–379773-X/$30.00

aboard the NASA Nimbus 7 satellite, the Earth Radiation Budget Satellite (ERBS), the Upper Atmosphere Research Satellite (UARS) and a series of meteorological satellites operated by the U.S. National Oceanic and Atmospheric Administration (NOAA). The result of this series of missions has been extensive knowledge of the distributions of trace constituents, particles, and temperatures in the earth's atmosphere. However, these measurements form only a part of an integrated observing strategy for atmospheric chemistry. Ground-, balloon-, and aircraft-based measurements are all needed to provide answers to the broad range of questions associated with atmospheric chemistry and transport.

The first space-based measurements of the trace distributions of stratospheric trace constituents (besides ozone) came from the Limb Infrared Monitor of the Stratosphere (LIMS) and Stratosphere and Mesosphere Sounder (SAMS) instruments aboard Nimbus 7. The LIMS instrument measured distributions of H_2O, NO_2, and HNO_3 (in addition to O_3 and temperature) for a period of 7 mo (10/78–5/79), while SAMS measured distributions of N_2O and CH_4 in the middle and upper stratosphere and lower mesosphere for approximately 3 yr. Both data sets had a significant impact on our understanding of chemical and dynamical properties of the stratosphere, especially on the interplay of the two processes. Measurements of NO_2, as well as of H_2O, were made by other instruments (SME, SAGE I, SAGE II).

The first instrument that allowed the global distribution of a trace gas to be measured is the *Total Ozone Mapping Spectrometer* (TOMS, http://jwocky.gsfc.nasa.gov). It retrieves information on total columns of ozone by comparing direct solar radiation with backscattered solar radiation at six wavelengths. Its spectral resolution is thought too low to measure trace gases other than ozone.

This was first overcome by the *Global Ozone Monitoring Experiment* (GOME, http://earth1.esrin.esa.it/eeo4.96) in 1995. With a total of 4096 channels it is the first instrument in space capable of measuring total columns of different trace gases from their absorption features on a global scale.

This chapter deals with the spectroscopic detection and analysis of biomass burning plumes and industrial pollution sources from GOME satellite data by means of *hyperspectral imaging*. We will present the analysis algorithms using the analysis of NO_2 as an example. Among the different trace gases emitted by mankind, *nitrogen oxides* play an important role. The natural concentration of NO_2 is believed to be lower than 10-20 ppt in the troposphere, whereas today concentrations as high as 200 ppb are found in cities. Natural sources of the nitrogen oxides are thunderstorms, bacteria in the soil, and bush and forest fires (*biomass burning*). Although NO_x[1] is toxic itself, its influence on atmospheric ozone chemistry makes it most important. Current estimates of the anthropogenic sources show high uncertainties of the global emission rates.

One of the major difficulties in the analysis is the time-consuming algorithm that calculates the trace gas concentration from optical satellite spectra. A new and significantly faster retrieval algorithm following the Differential Optical Absorption Spectroscopy (DOAS) concept will be presented, which allows the incoming satellite data to be processed in real time. In order to generate smooth concentration images it has to be taken into account that GOME orbits adjacent in time are not adjacent in space. Moreover, some pixels in an orbit may be missing, for example, due to satellite calibration or defective data retrieval. Efficient methods to interpolate concentration image sequences with a minimum loss of detail are described combining both a transport model for interpolation in time and normalized convolution for interpolation in space. Furthermore, cloud detecting algorithms have been developed to determine vertical columns of NO_2 concentrations. Additional electronic material can be found in /movies/37/gome.html.

37.2 The global ozone monitoring experiment (GOME)

The *Global Ozone Monitoring Experiment* (GOME) was launched aboard the European Space Agency's *Earth Resources Satellite* (ERS-2, http://earth1.esrin.esa.it/ERS) in 1995. GOME measures total column ozone, ozone vertical profile, and total column of several trace constituents, including global BrO, NO_2 and OClO as well as information on clouds, aerosols, and surface spectral reflectance. GOME can measure the ozone column at higher horizontal resolution than TOMS, and, thus, complements the TOMS observations. GOME covers a much broader wavelength range and has excellent spectral resolution (0.2–0.4 nm). In its polar sun-synchronous orbit it is well-suited for these measurements. With its high spatial and spectral resolution GOME is the first instrument in space that allows different stratospheric and tropospheric trace gases to be measured simultaneously from space on a global scale over a long period of time.

GOME consists of a set of four high-resolution absorption spectrometers designed to measure solar radiation scattered in the atmosphere. Each spectrometer has a diode array detector in its focal plane covering a spectral range from 290 to 790 nm with a spectral resolution of a total of 4096 pixels. Applying the concept of DOAS [1, 2], which is described in Section 37.3.1, information about trace gases in the atmosphere (for example, NO_x, O_3, O_4, BrO, OClO) can be obtained by analyzing the absorption features of each trace gas.

[1] $NO_x \Leftrightarrow NO + NO_2$

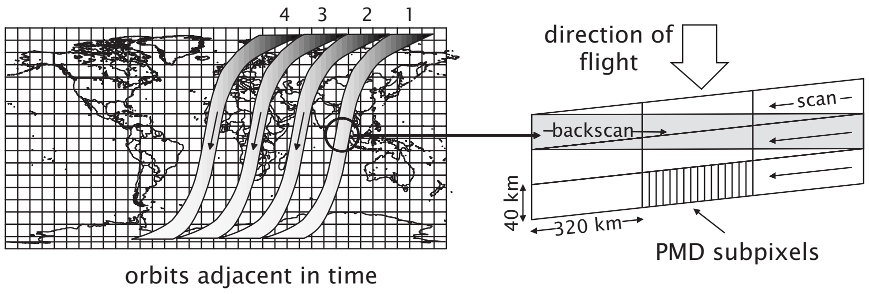


Figure 37.1: *The GOME spectrometer looks perpendicular to the earth's surface and scans a strip of three pixels with a spatial resolution of 320 × 40 km perpendicular to its direction of flight. It covers a range from 290 to 790 nm with a spectral resolution of 4096 pixels. The orbits adjacent in time are not adjacent in space but leave a gap of the size of two orbits. Only every three days is global coverage of the total surface achieved.*

37.2.1 Setup

The GOME instrument monitors the earth in nadir view (perpendicular to the earth's surface) and records an absorption spectrum every 1.5 s. Using a scanning mirror the instrument records the earth shine radiance (see Table 2.1) of three pixels perpendicular to the direction of flight and one backscan pixel at the end of the scanning period (see Figs. 37.1 and 37.2). In addition, the *Polarization Monitoring Device* (PMD) [3] simultaneously scans the earth with a higher spatial but lower spectral resolution. It gains information for 16 subpixels in each GOME pixel for an integral over three spectral bands between 295 and 745 nm (see Fig. 37.1). This subpixel information can be used for cloud detection as described in Section 37.5.2.

To control spatial resolution the width of the GOME pixels can be modified, by changing the maximum tilt angle of the scan mirror, to 120 km, 240 km, 320 km, 480 km and 960 km, whereas their height is constantly 40 km. Only for pixels larger or equal than 320 km (global measurement mode) the pixels are adjacent and global coverage of the earth's surface is achieved. For the other cases an image formation algorithm has to fill in the gaps caused by nonadjacent pixels or missing pixels due to defective data transfer (see Section 37.4). The satellite scans the earth in consecutive orbits that are spatially not adjacent to each other but leave a gap that is the size of two orbits. So after cycling the earth for a single day (14 orbits) it has covered only 1/3 of its surface and scans on the following day the left adjacent orbit. Only every three days is total coverage obtained.

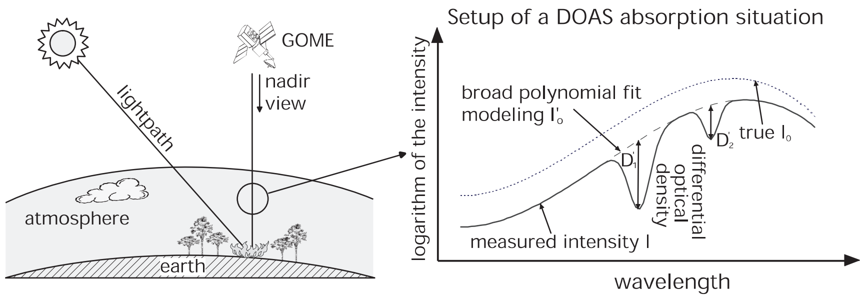

GOME
nadir view
lightpath
atmosphere
earth
Setup of a DOAS absorption situation
logarithm of the intensity
broad polynomial fit modeling I'₀
true I₀
D'₁
D'₂
differential optical density
measured intensity I
wavelength

Figure 37.2: *Setup of the GOME viewing geometry. The light emitted by the sun is attenuated by the atmosphere along the lightpath l before being scattered and reflected into the satellite's spectrograph. As the absorption structures are characteristic for each absorbing tracer their slant column densities (lightpath $\times$ concentration) may be calculated using the DOAS algorithm. For the calculation of the effective light path the knowledge of clouds is essential.*

37.2.2 Earth coverage

The viewing mode of GOME results in images constructed as a mean over three days consisting of portions taken at different times, comparable to interlaced TV images. To analyze motion of air masses temporal interpolation on the basis of a simple transport model has been applied (see Section 37.4). For this purpose third party wind field information has been used to estimate the concentration at points in space and time not covered by the satellite instrument.

37.3 Retrieval of trace gas concentrations

The first and most important step in the analysis of the spectra delivered by the satellite is the calculation of the concentrations of trace gases in the atmosphere. For this purpose the *differential optical absorption spectroscopy* (DOAS) technique[2], first introduced by Noxon [4] and Platt et al. [5], allows the amount of trace gases along the light path to be calculated from the absorption cross section of each trace gas.

37.3.1 Differential optical absorption spectroscopy (DOAS)

For the instrumental setup of the GOME instrument, light with an initial intensity $I_0(\lambda)$ is emitted by the sun, passes a layer of absorbing species (atmosphere) and is collected by the GOME spectrograph (see Fig. 37.2). During its way through the atmosphere the light undergoes extinction due to absorption processed by different trace gases and scattering by

[2]see http://aguas1.uphys.uni-heidelberg.de/urmel/project.html

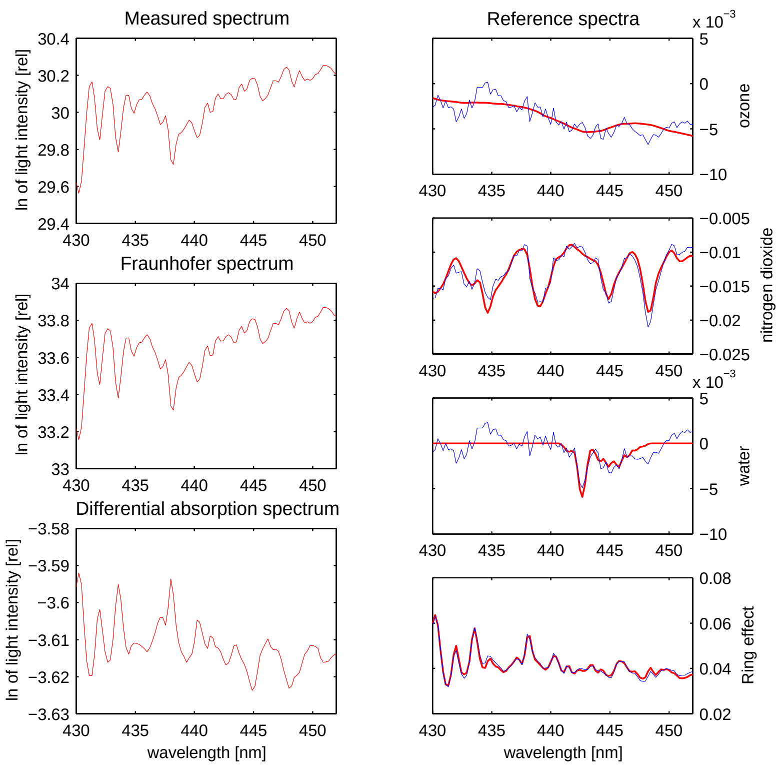


Figure 37.3: *Example for the evaluation of a GOME spectrum. The main absorption structures are those of the Fraunhofer lines. After the removal of these structures a differential structure remains, which contains the spectral information for the trace gases (left). This structure is decomposed into the proportions of the reference spectra applying the algorithms presented in the text (right).*

air molecules and aerosol particles. The measured intensity $I(\lambda)$ at the instrument is, therefore, given by *Lambert-Beer's law* Eq. (3.27):

$$I(\lambda) = I_0(\lambda) \exp\left(-\sum_k \sigma_k(\lambda) S_k\right) A(\lambda) \tag{37.1}$$

The sum in the exponential runs over all trace gases i in the lightpath l whose *slant column density* $S_k = \int c_k(l)\,\mathrm{d}l$ can be calculated. The absorption cross sections $\sigma_k(\lambda)$ are well known and they are characteristic of each trace gas. The factor $A(\lambda)$ describes additional attenuation by the optical system and by *Rayleigh* and *Mie* scattering (see Volume 1, Section 3.4.1, Reqshhrmthhrmt) at aerosols in the atmosphere. An example of this situation can be seen in Fig. 37.3.

So far, Eq. (37.1) can be transformed into a linear equation with respect to S_k by taking the logarithm of both sides:

$$\ln I(\lambda) = \ln I_0(\lambda) + \ln A(\lambda) - \sum_k \sigma_k(\lambda) S_k \tag{37.2}$$

For the evaluation it must be taken into account that the initial light intensity $I_0(\lambda)$ and the attenuation effect $A(\lambda)$ are unknowns while only the absorption cross sections $\sigma_k(\lambda)$ are known from measurements in the laboratory. $I(\lambda)$ is the measured quantity from the spectrometer. The essential observation that leads to DOAS and allows S_k to be calculated without the knowledge of $I_0(\lambda)$ and $A(\lambda)$ is that those quantities contain only broad spectral structures with respect to λ whereas the absorption cross sections $\sigma_k(\lambda)$ contain both broad and narrow proportions. The later can, thus, be split into such contributions $\sigma_k = \sigma_k^{\mathrm{brd}} + \sigma_k^{\mathrm{nrw}}$. If all broad proportions are modeled by a polynomial

$$p(\lambda) \equiv \sum_j a_j \lambda^j \approx \sum_k \sigma_k^{\mathrm{brd}}(\lambda) + \ln A(\lambda) + \ln I_0(\lambda) \tag{37.3}$$

then Eq. (37.2) becomes

$$\ln I(\lambda) = \sum_j a_j \lambda^j - \sum_k \sigma_k^{\mathrm{nrw}}(\lambda) S_k \tag{37.4}$$

which is linear in the unknown quantities a_j and S_k and can be solved applying a linear least squares method.

During the recording procedure of the GOME instrument the incoming light intensity $I(\lambda)$ is sampled at n discrete pixels i by the diode array in the spectrograph. The continuous wavelength λ is, therefore, mapped to discrete pixels. This is described by a mapping function $\Lambda : \mathcal{N} \to \mathcal{R}$. In general the function can be sufficiently approximated by a polynomial

$$\lambda_i \equiv \Lambda(i) = \sum_l b_l i^l \tag{37.5}$$

The coefficient b_0 describes the offset of the wavelength interval recorded by the spectrograph, the coefficient b_1 its linear dispersion relation.

However, it emerges that an additional effect has to be taken into account: it is observed that the wavelength mapping of the spectrograph does not stay constant over time but changes, for example, due to thermal dilation. It is thus not sufficient to calibrate the wavelength mapping once but this has to be done for each spectrum in an orbit. The equation that has to be solved transforms, thus, from Eq. (37.4) applying Eq. (37.5) to

$$\ln I_i = \sum_j a_j \lambda_i^j - \sum_k \sigma_k^{\mathrm{nrw}}(\lambda_i) S_k = \sum_j a' i^j - \sum_k \sigma_k^{\mathrm{nrw}}\Big(\sum_l b_l^k i^l\Big) S_k \tag{37.6}$$

In this nonlinear equation the quantity I_i denotes the intensity measured at pixel i of the spectrograph. The broad polynomial $\sum_j a_j \lambda_i^j$ has been merged with the polynomial describing the mapping function Λ to become a polynomial with the new coefficients a'. The index k at the mapping coefficient b_l^k points out that the dispersion relation may be different for each absorbing species k and has, thus, to be calculated independently.

The following section concentrates on the solution of the nonlinear problem Eq. (37.6) in an efficient way, as in contrast to the standard approach we have to deal with a large number of data spectra. In the case of GOME the algorithm has to be able to deal with approximately 20,000 spectra each day, which have to be processed in real time. The presented algorithms are able to achieve this on standard PC hardware (Pentium Pro 200 Mhz).

The major difficulty lies in the large number of unknown parameters (typically between 20 and 30) as the coefficients of the broad polynomial a as well as the mapping parameters b and the slant column densities S have to be calculated. The other time-consuming step originates from the fact that the absorption cross sections σ are on their part discretely sampled functions. As Eq. (37.6) requires them to be evaluated at arbitrary intermediate grid points an interpolation algorithm has to be applied in each fitting step. Section 37.3.2 thus deals with

- the adaptation of an efficient fitting algorithm [6] taking into account that most of the parameters in Eq. (37.4) are linear and only the parameters b describing the dispersion relation are nonlinear, and
- the application of a fast interpolation algorithm on the basis of B-splines [7] that combines both interpolation accuracy and speed.

37.3.2 Nonlinear fitting algorithm

The problem of solving an equation like Eq. (37.6) is not unique to the DOAS problem but arises in many other cases. The equation will, thus, be generalized and rewritten in matrix notation

$$\boldsymbol{L}(i) = \boldsymbol{R}(\boldsymbol{b}, i)\boldsymbol{a} \tag{37.7}$$

In this equation the vector $\boldsymbol{L}$ denotes the vector of the logarithm of the measured intensities $\boldsymbol{L} = [\ln I(\lambda_0), \ldots, \ln I(\lambda_{n-1})]^T$ at the n discrete channels i. The matrix $\boldsymbol{R}$ is defined as a combination of the l reference spectra σ and the $m+1$ basis functions of the polynomial modeling the broad absorption structures. It is called the *design matrix* of the problem:

$$\boldsymbol{R}_{j,i}(\lambda) = \left[\boldsymbol{\lambda}^0, \ \ldots, \ \boldsymbol{\lambda}^m, \boldsymbol{\sigma}_0^{\mathrm{nrw}}, \ldots, \boldsymbol{\sigma}_{l-1}^{\mathrm{nrw}}\right]^T \tag{37.8}$$

The linear parameters are gathered in the vector

$$\boldsymbol{a} = [a_0, \ldots, a_m, S_0, \ldots, S_{l-1}]^T \tag{37.9}$$

whereas the nonlinear parameters are mapped to the vector $\boldsymbol{b}$ with $\boldsymbol{b}_{l*i+j} = b_i^j$ compared to the notation in Eq. (37.6).

In order to minimize the total number of fit parameters in the nonlinear fit of Eq. (37.6) an algorithm from Ottoy and Vansteenkiste [6] has been adapted. Their algorithm is optimized for problems in which linear and nonlinear parameters can be separated. The direct approach attempts to solve Eq. (37.7) for the unknown parameters $\boldsymbol{a}$ and $\boldsymbol{b}$ by minimizing the equation

$$|\boldsymbol{L} - \boldsymbol{R}(\boldsymbol{b}, i)\boldsymbol{a}|^2 \Rightarrow \min \tag{37.10}$$

iterating both on $\boldsymbol{a}$ and $\boldsymbol{b}$. Ottoy and Vansteenkiste [6] show that Eq. (37.10) can be solved alternatively by first solving the equation for the linear parameters $\boldsymbol{a}$ under the assumption that $\boldsymbol{b}$ is constant, yielding $\boldsymbol{a} = \left(\boldsymbol{R}^T\boldsymbol{R}\right)^{-1}\boldsymbol{R}^T\boldsymbol{L}$. Substituting $\boldsymbol{a}$ in Eq. (37.10) and minimizing the resulting equation

$$|\boldsymbol{L} - \boldsymbol{R}\left(\boldsymbol{R}^T\boldsymbol{R}\right)^{-1}\boldsymbol{R}^T\boldsymbol{L}|^2 \Rightarrow \min \tag{37.11}$$

is then equivalent to the original minimization problem yielding the advantage that the linear parameters are fitted implicitly and iteration has only to be done on the remaining nonlinear parameters $\boldsymbol{b}$.

This means an effective reduction of fitting parameters by 30 %, resulting in faster convergence and less computational effort for each fitting step compared with the direct method.

37.3.3 B-spline interpolation

During the fitting process it is necessary to evaluate the absorption cross sections $\sigma(\lambda)$ at arbitrary wavelengths λ due to the unknown dispersion relation in Eq. (37.6). As the absorption cross sections are only known as discretely sampled spectra this implies the necessity for an interpolation algorithm.

The most qualified interpolation algorithm is the *B-spline interpolation* (see Volume 2, Section 8.5 for details). B-splines of order n, $\beta^n(\lambda)$, are piecewise continuous polynomial functions that form a base for all piecewise continuous polynomial functions. The main characteristic of B-spline interpolated functions is the continuity of the derivatives up to order $n-1$ at the grid points. This feature is achieved by generating the spline functions as an n-fold convolution of the box function. Analyzing the transfer function one realizes that the *interpolation condition* (see Volume 2, Section 8.2.3) is not fulfilled and so a transformation is required. With this transformation and the following convolution the signal may, thus, be decomposed in B-splines and so be evaluated at arbitrary wavelengths according to the following equation

$$\sigma^{\text{nrw}}(\lambda) = \sum_{i=-\infty}^{\infty} c_i \beta^n(\lambda - \lambda_i) \tag{37.12}$$

Unser et al. [7] showed that the decomposition can be done very efficiently using a recursive filter scheme and also describes an algorithm that calculates the evaluation at intermediate grid points with only 4 additions and 4 multiplications per data point for B-splines of order 3. As an additional advantage, once the decomposition has been performed the derivatives of the signal—that are required during the fitting algorithm—with respect to λ can be computed directly using forward differences on the decomposition coefficients c_i [8].

37.4 Image interpolation

During data acquisition the satellite scans the earth in orbits that are not adjacent in space but leave a gap of two orbits every day (see Section 37.2.2).

To gain full concentration maps for each day it is, thus, necessary to fill in the missing pixels using an appropriate interpolation technique. As the concentrations are not static in the atmosphere, wind vector information obtained from independent sources has been used to model the transport and to interpolate the missing data.

37.4.1 Interpolation using a wind-vector field

For nonstatic NO_x sources, it is evident that information about the wind is required in order to calculate the concentration in the gaps. The wind vector fields are taken from NILU's Atmospheric Database for Interactive Retrieval (NADIR, `http://www.nilu.no/first-e.html`) in Sweden and have been calculated by an ECWMF (`http://www.ecmwf.int`) weather prediction model. The wind data is subdivided in different pressure levels and because the measured NO_x data is not height-resolved the pressure level that is best suited for our causes has to be selected. We use the 750 hPa pressure level corresponding to 2.5 km altitude because the NO_x plumes are typically located in this area [9, p. 23 and 522].

In Fig. 37.4 the vertical distribution of ozone is shown. High concentration of NO_x emitted from biomass burning events leads to a strong atmospheric ozone enhancement by photochemical production considering a short timescale and also an accumulation of stratospheric ozone

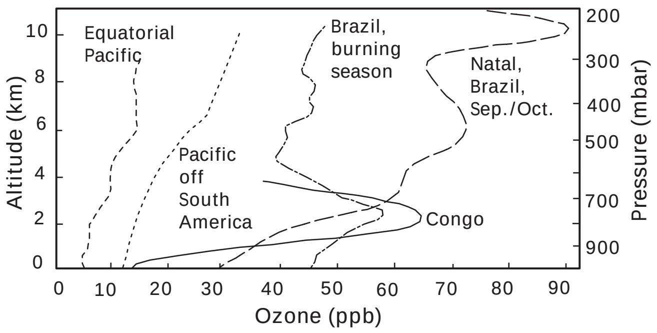


Figure 37.4: *Summary of observed ozone profiles in the equatorial zone [9, p. 23,522]*

considering a large timescale. As our interest is focused on the atmospheric NO_x we use the typical altitude for biomass burning plumes indicated by the strong ozone enrichment shown in Fig. 37.4.

The intrinsic resolution of the wind vector data is 1.125° in space and 6 h in time, resampled by linear interpolation to the required resolution. Typical for transport in the atmosphere is its highly turbulent flow with divergences. The equation of motion is:

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{u}(x, y, t) \tag{37.13}$$

with the given windspeed vector fields $\boldsymbol{u}$. This equation is not analytically solvable because $\boldsymbol{u}$ is not an analytic function. Therefore, finite differences are used to model the distortion.

We divide the picture in small triangles with the extension of 2×2 pixel and calculate the new position of the corners applying

$$\boldsymbol{x}(t + \Delta t) = \boldsymbol{x}(t) + \boldsymbol{u}(x, y, t)\Delta t \tag{37.14}$$

using forward difference.

The trajectories of the corners are calculated in one hour time steps for one day. When the positions of all corners are calculated they will again be discretized and stored in the image. The distortion of the triangles is shown in Fig. 37.5.

If the triangle expands the added pixels are linearly interpolated (bilinear patch) and are scaled down because the sum of concentrations must remain constant regardless of the decay of the NO_x. Analogously, if the triangle is compressed the concentrations are scaled up. The

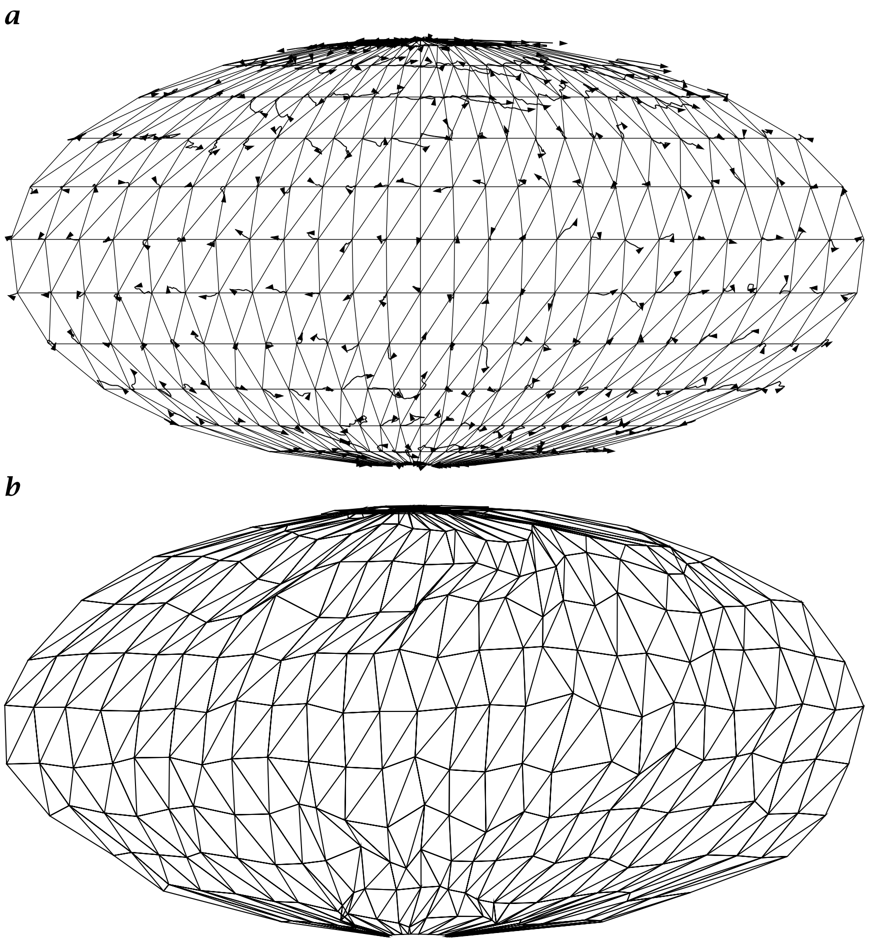


Figure 37.5: ***a*** *Dissection of the image in triangles, the scale of the triangles is increased for better oversight. For the purpose of visualization the Mollweide projection is used. Additionally the trajectories of the corners are shown.* ***b*** *Distortion of the same image after 10 days.*

integral of the concentrations over the new triangle is then given by

$$\int_{\Delta} c(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \frac{1}{6}(c_1 + c_2 + c_3)\,(x_1(y_3 - y_2) + x_2(y_3 - y_1) + x_3(y_2 - y_1)) \tag{37.15}$$

which leads to a scaling factor for the transported concentrations

$$f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \frac{6}{x_1(y_3 - y_2) + x_2(y_3 - y_1) + x_3(y_2 - y_1)} \tag{37.16}$$

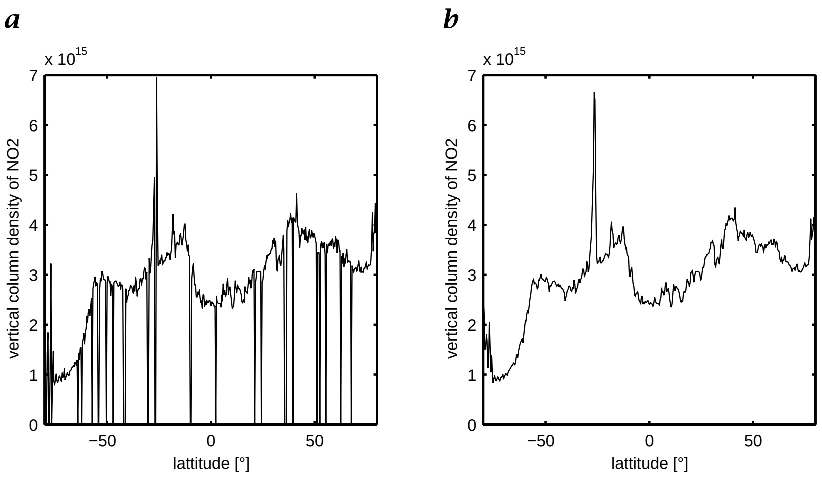


Figure 37.6: *Example for the result of the normalized convolution.* ***a*** *shows a slice through an image of vertical NO_2 concentrations in latitudinal direction after the transport step. There still remain missing irregularly distributed pixels. After the application of normalized convolution in* ***b*** *(see text) the missing pixels are interpolated without blurring the available information.*

The reduction of the concentrations caused by the chemical decay must be calculated separately. With this method, we get a full one day image out of the data of the last few days and can use this image to fill in the missing orbits of the following pictures.

37.4.2 Normalized convolution

Subsequent to the transport step there still remain small gaps in the images arising from defective data retrieval by the satellite (see Fig. 37.6). These gaps are distributed irregularly over the image and thus can not be interpolated using a B-spline interpolation algorithm which relies on information available on a regular grid. Instead the images are interpolated using *normalized convolution* [10] (see also Volume 2, Section 7.6 for a detailed description).

For each concentration image g a mask image m is constructed that contains 1 for pixels containing information and 0 else. The interpolation is then performed by applying an *applicability function* to both the image and the mask. In the following normalization step the image is divided by the mask. The applicability function behaves like a smoothing filter that allows information to diffuse from the neighborhood pixels into regions with missing or poor information. The operation can, therefore, be written as

$$g_{\text{ipol}} = \frac{b * (g\, m)}{b * m} \tag{37.17}$$

with the applicability function b. Knutsson and Westin [10] use for analytical reasons an isotropic function of the form

$$b(|x|) = \begin{cases} |x|^{-\alpha} \cos^{\beta}(\dfrac{\pi |x|}{2|x|_{\max}}) & |x| < |x|_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{37.18}$$

However this function has the disadvantage of being nonseparable, which leads to a computationally ineffective implementation. As our application deals with large data sets we apply a different function as a filter kernel. For the sake of separability we disclaim the isotropy and apply a recursive filter proposed by Deriche [11]. The filter is steerable by a parameter α, which describes the width of the filter kernel, and the radius of influence in the neighborhood of a pixel.

$$b(x) = k(\alpha|x| + 1)e^{-\alpha|x|} \text{ with } k = \frac{(1 - e^{-\alpha})^2}{1 + 2\alpha e^{-\alpha} - e^{-2\alpha}} \tag{37.19}$$

For a large value of α (which denotes only little influence of the neighborhood pixels) this filter approximates the filter Eq. (37.18) very well. The advantage of the usage of a recursive filter lies in the fact that the computational effort does not increase with a growing area of influence as in the case of nonrecursive filters. Even for massive blurring ($\alpha < 1$) we only need 8 multiplications and 7 additions per data point to perform the normalization.

An example for the normalization step is shown in Fig. 37.6, where a slice of a NO_2 map is displayed together with the interpolated result ($\alpha = 3$). It can be seen that even the fine structures of the signal are preserved whereas all gaps of different sizes could be interpolated.

37.5 Determination of vertical columns

37.5.1 Air mass factors

So far the DOAS algorithm in Section 37.3.1 calculates *slant column densities* (SCD) of the trace gases in question. The resulting values of the solution of Eq. (37.4) S_i hence denote the integral along all possible light paths over the concentration of a trace gas (see Fig. 37.2).

For further analysis the slant column densities S_i calculated by the DOAS algorithms must be transformed into *vertical column densities* (VCDor V_i). Both quantities only differ by the *air mass factor* (AMF) or

A), which describes the effective path length of the light that is emitted by the sun and falls into the spectrometer:

$$V_i = \frac{S_i}{A} \tag{37.20}$$

Due to scattering effects in the atmosphere a huge distribution of possible paths exists that makes the calculation of the AMF computationally very expensive. For the calculation of the AMF we employ the program AMFTRAN, which was developed at the University of Heidelberg. It uses the results of the cloud detection algorithm formerly described as input quantities. Its calculations are based on radiation transport and it considers multiple scattering effects to estimate the effective path length. For details see Marquard [12].

37.5.2 Cloud detection

One of the most important parameters for the calculation of the AMF is the cloud coverage at the measurement pixel. The pathlength of the light in a cloudy and cloud free environment differ in a wide range. To calculate the length of the aabsorption path it is, thus, necessary to determine whether the light beam is reflected at the earth's surface or at a cloudy layer. Therefore, a cloud detecting algorithm is needed.

The cloud detection algorithm presented in this section relies on information from the spatially high resolution PMD data (see Section 37.2). These data yield an integral of the light intensity over the three wavelength intervals (see Volume 1, Fig. 2.1) 295 to 397 nm (UV), 397 to 580 nm (visible) and 580 to 745 nm (red), and will, thus, be regarded as RGB values for further image processing. The basic idea for the cloud detecting algorithms is to concentrate on two characteristics of clouds, their degree of whiteness and the fact that they form a moving layer in front of a static background.

The HSV Method. A characteristic attribute of clouds is their degree of whiteness compared to the background pixels (see Fig. 37.9b). The appropriate color model to measure whiteness is the *HSV color model* [13]. In this model the RGB color space is transformed to a cylindrical coordinate system, which is similar to the IHS color coordinate system described in Volume 1, Section 11.3.5. The H denotes *hue* and is measured by the angle around the vertical axis, S the *saturation* of the color relative to the color gamut and V specifies the brightness value. A 3-D animation of the HSV-cone can be seen in (.) We now can define a subset of space within the pyramid which characterizes the clouds. In order to do this it is necessary to take a look at a histogram plot over saturation and value calculated from a time series of PMD pictures. The histogram plot for the PMD data of 30 days is shown in Fig. 37.7b. There we can

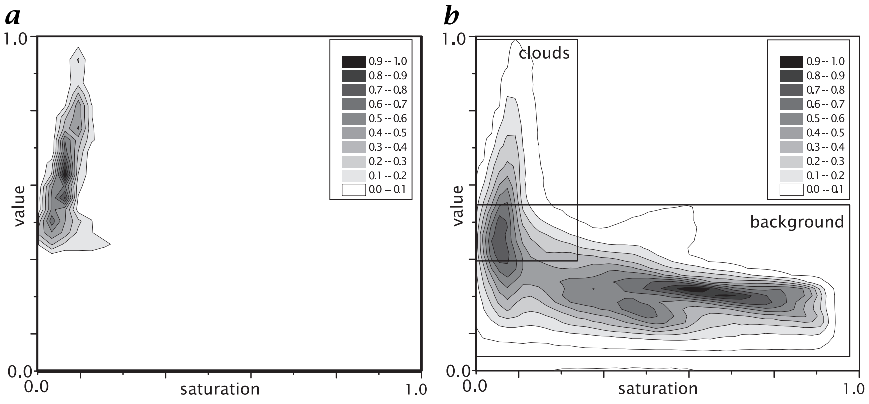


***Figure 37.7:** **a** Histogram of the distribution of saturation (S) and value (V) in a sequence of PMD images in which the clouds were cut out manually; **b** histogram of the same sequence of full PMD images. The regions classifying clouds and background are marked by rectangles. It can be seen that they only slightly overlap so segmentation of the clouds can be successfully performed applying a threshold.*

see that the regions classifying the background have only a small overlap with the region of clouds. Thus, efficient detection can be achieved by applying a threshold in the S-V color space. This classifying region for clouds is obtained by analyzing an amount of pixels which is sorted out as clouds manually, see Fig. 37.7a, and is confirmed by the second, iterative cloud detection method, herein later described. Though the results are very convincing, the method determines only whether or not there is a cloud but does not calculate the degree of cloud coverage. Moreover, the limiting values for the cloud subspace are in a way arbitrary and disregard local conditions such as ground albedo.

Iterative method. The results of the HSV method can be improved by considering that clouds are moving, forming and dissolving. Therefore, those values of a time series that stay nearly constant are likely to belong to the background whereas those that change should belong to clouds. This approach is successful if the majority of days are cloud free. The implementation of this idea is done in an iterative algorithm.

- First we calculate the mean RGB values over a series of pictures. This average picture serves as a first estimate for the background picture.
- This background picture can then be used in the second step to measure the deviation of each pixel. For this purpose a weighting function $w(\|\boldsymbol{A}_k(\boldsymbol{x}) - \boldsymbol{B}^n(\boldsymbol{x})\|)$ is required, which is 1 if the actual

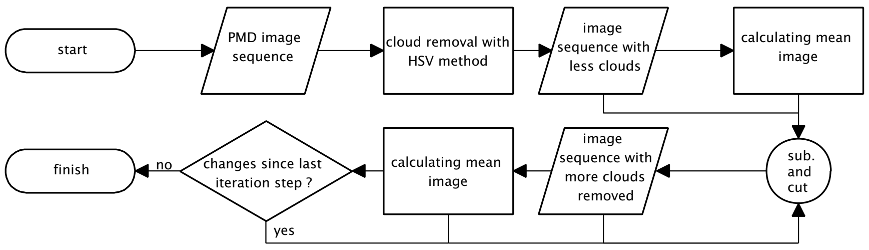

start
PMD image sequence
cloud removal with HSV method
image sequence with less clouds
calculating mean image
sub. and cut
image sequence with more clouds removed
calculating mean image
changes since last iteration step ?
no
yes
finish

Figure 37.8: *Flow chart of the cloud-detecting method. In the first step the clouds are removed with the HSV method. Then each image is compared with the mean over all images and pixels with high deviation are removed. This step is iterated until the mean image remain constant.*

pixel $\boldsymbol{A}(\boldsymbol{x})$ is the same as the one from the background image $\boldsymbol{B}(\boldsymbol{x})$, and 0 if there is a high deviation referring to the three PMD values.

- In the third step the weighted mean over the picture series is calculated, using the measuring function w. Now yielding a better estimate for the background picture, we go back to the second step.
- When the background picture stays nearly constant, the algorithm terminates.

If $\boldsymbol{B}^n(\boldsymbol{x})$ is the background picture of the n^{th} iteration and $\boldsymbol{A}_k$ the k^{th} picture of the time series, we have to find the fix-point of the following function:

$$f(\boldsymbol{B}^n(\boldsymbol{x})) = \boldsymbol{B}^{n+1}(\boldsymbol{x}) = \frac{\sum_k^K \boldsymbol{A}_k(\boldsymbol{x}) w(\|\boldsymbol{A}_k(\boldsymbol{x}) - \boldsymbol{B}^n(\boldsymbol{x})\|)}{\sum_k^K w(\|\boldsymbol{A}_k(\boldsymbol{x}) - \boldsymbol{B}^n(\boldsymbol{x})\|)} \tag{37.21}$$

$w(\|\boldsymbol{A}_k(\boldsymbol{x}) - \boldsymbol{B}^\infty(\boldsymbol{x})\|)$ is used to calculate a degree of cloud coverage.

After testing different weighting functions w we found that the one yielding the fastest convergence is the step function Θ of the Euclidean distance in the S-V color space, where the threshold is set to the quadratic standard deviation $sd(\boldsymbol{x})$.

$$w(\boldsymbol{A}(\boldsymbol{x})) = \Theta(sd(\boldsymbol{x}) - \|\boldsymbol{A}(\boldsymbol{x})\|) \text{ with } \boldsymbol{A}(\boldsymbol{x}) = [A_S, A_V]^T \tag{37.22}$$

This function requires the least iterations and the results of cloud detection are acceptable.

Both described algorithms have their strengths and weaknesses. The first algorithm is very fast whereas the second algorithm is used to improve the results of the first one. Moreover, it can be better adapted to different times of the year simply by calculating different background

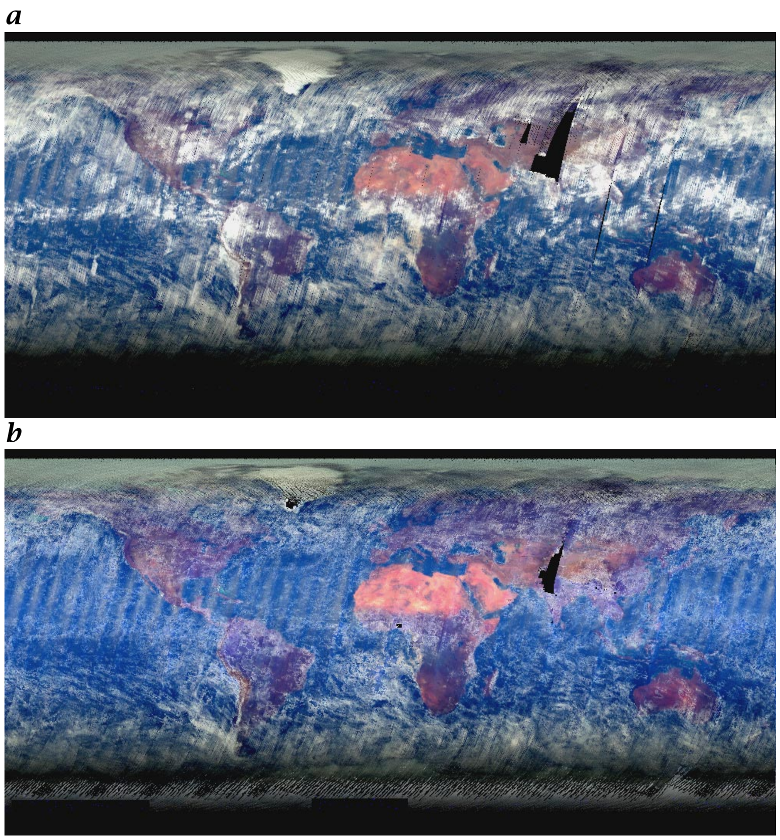


Figure 37.9: ***a*** *Mean image over a time series of PMD images over 30 days. The clouds can be easily distinguished with their whiteness.* ***b*** *Mean image after cloud removal. Most of the clouds could be located and removed to obtain a cloud free background image; (see also Plate 13).*

pictures for each season, so that we can distinguish, for example, between clouds and snow. Hence, a combination of the two described methods is the most efficient approach (see Fig. 37.8). The results of the cloud detecting method are shown in Fig. 37.9.

37.5.3 Separation of tropospheric and stratospheric gas concentrations

Hitherto the analysis procedure has been able to calculate the amount of trace gases integrated along a vertical column. Evidently the quantity contains both the stratospheric and the tropospheric proportion of the trace gas in question.

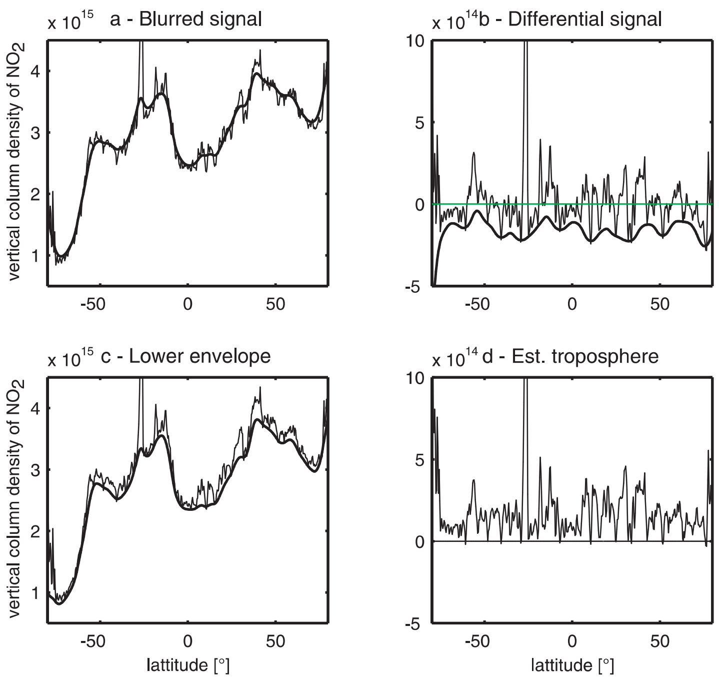


Figure 37.10: *Necessary steps in the separation of stratosphere and troposphere.* ***a*** *shows the original signal together with the blurred signal as an estimate of the stratosphere. The differential signal* ***b*** *is partially negative so the lower envelope is found by an erosion operation (subimage* ***c****).* ***d*** *shows the resulting tropospheric amount.*

However, to estimate the total amount of NO_2 emitted by biomass burning plumes only the tropospheric proportion is of interest. This is because the pollutant is emitted in the lower troposphere and accumulates at an altitude of approximately 2.5 km (see Section 37.4.1).

For the separation of the two proportions we exploit the fact that the distribution of the stratospheric NO_2 is much more homogeneous than its distribution in the troposphere (see Fig. 37.10). Due to strong stratospheric winds and mixing processes the stratospheric NO_2 distribution contains only low spatial frequencies. In the troposphere the total NO_2 concentration is approximated by a factor 10 lower than in the stratosphere but as the sources of NO_2 in the troposphere are localized its distribution shows much stronger variances. This can be used for separation by a multilevel algorithm (see Fig. 37.11):

- First the concentration image g is decomposed by a bandpass filter into an image t that contains only spatial frequencies in the order of

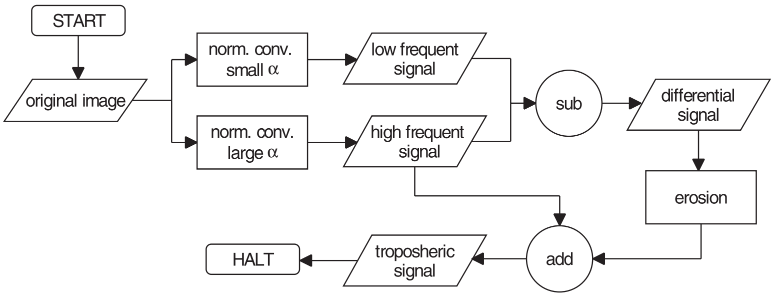


***Figure 37.11:** Separation of the stratospheric and tropospheric concentration. A bandpass separates the high frequent tropospheric structures from the stratospheric background. As the troposphere may only be non-negative an erosion filter is applied to the differential signal to determine its lower envelope (see also Fig. 37.10).*

magnitude of the expected size of the biomass burning plumes. The remaining image $s = g - t$ gives a first estimate for the stratospheric background. As filters we use *Deriche filters* of type Eq. (37.19) with $\alpha = 3$ for the high cutoff frequency and $\alpha = 1$ for the low cutoff frequency.

- The properties of the bandpass filter cause the estimated background s to yield a mean value of the concentration distribution, which leads to negative values for t. As t is only supposed to increase the stratospheric background, s must be corrected by finding the lower envelope of t under the boundary condition that this signal should not contain higher frequencies than s. This is done by an *erosion* operation on the differential signal t. The mask for this operation is chosen such that its size N corresponds to the size of a binomial smoothing mask, which is equivalent to the applied Deriche filter: $N = 16/\alpha^2 = 16$.
- The differential signal is then corrected by the result of the erosion operation, which guaranties non-negative values for the tropospheric proportion.

The results of this operation can be seen in Fig. 37.10a, b. In contrast to a simple bandwidth filter the envelope that describes the stratospheric background is found such that it lies totally below the measured distribution and does not contain frequencies beyond a threshold.

37.6 Results

In this chapter algorithms were presented to calculate sequences of concentration maps of trace gases in the atmosphere such as NO_2 from

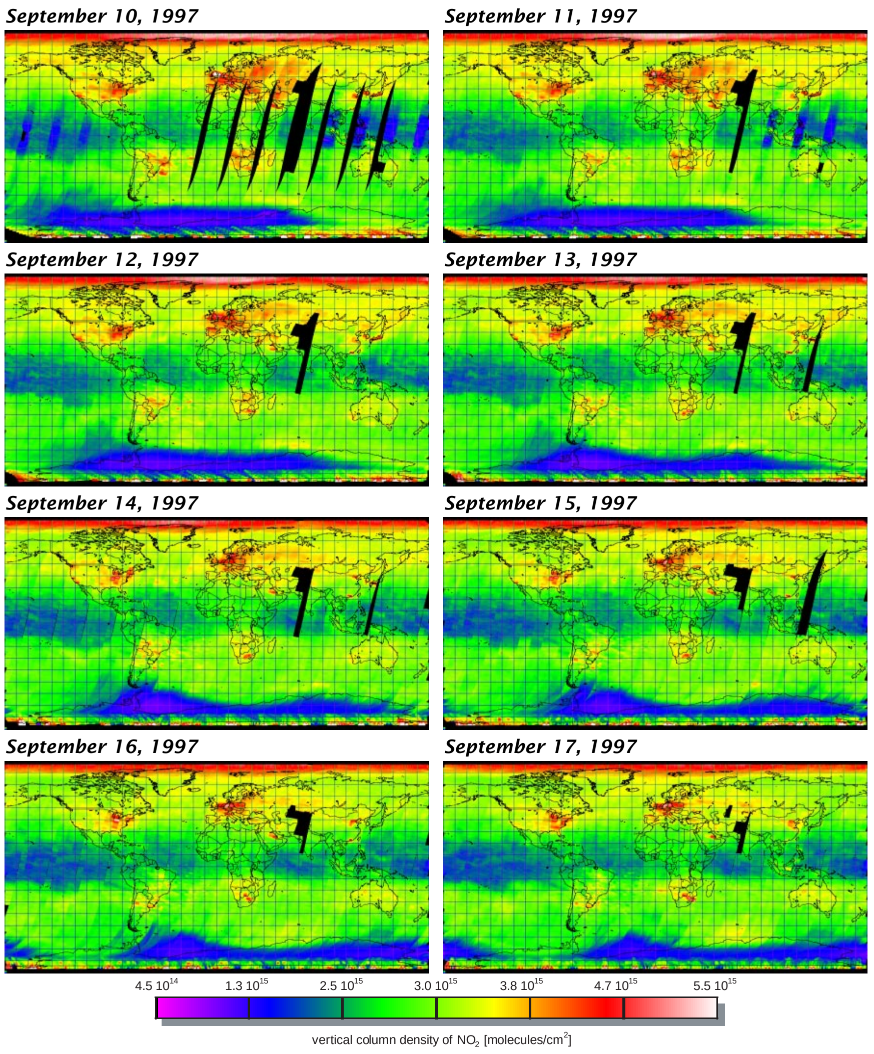


Figure 37.12: *Time sequence of vertical column densities of NO_2. Large emission plumes can be noticed both over industrialized areas in Europe and North America and over regions where biomass burning occurs (South America, Africa). The time sequence shows the movement of the plumes over a time period; for a movie covering the whole year 1997, see `/movies/37/no2.mov`; (see also Plate 14).*

raw spectra of the GOME satellite instrument. By the application of a new fitting algorithm and an improved interpolation technique on the basis of B-splines (Section 37.3.1) the retrieval algorithm could be accelerated by one order of magnitude compared to existing algorithms [2]. The evaluation of a full GOME orbit, consisting of approximately 2000 single spectra, now only takes 2 min on a standard pentium pro 200 computer. Thus the evaluation of total coverage consisting of 36 orbits every three days takes 1.2 hours and can, thus, be performed in real-time, leaving time to consecutively perform transport calculations and cloud detection.

By the application of transport calculations on the basis of finite differences full maps of vertical column densities of NO_2 could be generated with a resolution of one day. A time sequence of six days of such images is shown in Fig. 37.12. On these images highly elevated NO_2 concentrations can be observed both over the industrialized regions in North America and Europe and over poorly industrialized regions in South America and Africa. The emissions over South America and Africa are most likely due to biomass burning.

For the calculation of the absolute NO_x emission a cloud detection algorithm has been developed that combines both a threshold in the HSV-color space and an iterative fix-point algorithm that allows detection of clouds in highly as well as lightly cloudy regions (see Fig. 37.9b). This step and the separation of the tropospheric and stratospheric proportions of the vertical column density are the essential preconditions for further analysis. On the basis of this time sequence information it is now possible to calculate the total amount of the NO_x emission within these plumes on a global scale.

The knowledge of the wind vector field already used for interpolation purposes will allow tracking of plumes over time and estimating of the total NO_2 emitted in those plumes.

Acknowledgment

The authors thank the Deutsche Forschungsgemeinschaft (DFG, http://www.dfg.de) for financial support for this research within the research unit “Image Sequence Analysis to Investigate Dynamic Processes” (http://klimt.iwr.uni-heidelberg.de/PublicFG). C. Leue is grateful for a dissertation grant by the Studienstiftung des Deutschen Volkes (http://www.studienstiftung.de). M. Wenig thanks the Graduiertenkolleg “Modeling and Scientific Computing in Mathematics and Natural Science” (http://www.iwr.uni-heidelberg.de/~GK/GK_home.html) for a dissertation grant.

37.7 References

[1] Platt, U., (1994). Differential optical absorption spectroscopy (DOAS). In *Air Monitoring by Spectroscopic Techniques*, M. Sigrist, ed., Vol. 127. New York: John Wiley & Sons, Inc.

[2] Stutz, J. and Platt, U., (1996). Numerical analysis and error estimation of differential optical absorption spectroscopy measurements with least-squares methods. *Applied Optics*, **35**:6041–6053.

[3] Bednarz, F., (1995). *Global Ozone Monitoring Experiment, GOME, Users Manual.* Technical Report, ESA Publications Division.

[4] Noxon, J., (1975). Nitrogen dioxide in the stratosphere and troposphere measured by ground-based absorption spectroscopy. *Science*, **189**:547–549.

[5] Platt, U., Perner, D., and Pätz, (1979). Simultaneous measurements of atmospheric CH_2O, O_3 and NO_2 by differential optical absorption. *J. Geophys. Res.*, **84**:6329–6335.

[6] Ottoy, J. and Vansteenkiste, G., (1981). A computer algorithm for non-linear curve fitting. *Advances in Engineering Software*, **3**:55–61.

[7] Unser, M., Aldroubi, A., and Eden, M., (1991). Fast B-spline transforms for continuous image representation and interpolation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **13**(3):277–285.

[8] Jähne, B., (1997). *Digital Image Processing: Concepts, Algorithms and Scientific Applications.* Berlin, Heidelberg: Springer.

[9] Lindesay, J. A., Andreae, M., Goldhammer, J. G., Harris, G., Annegarn, M., H. J. Garstang, Scholes, R. J., and van Wilgen, B. W., (1996). International geosphere-niosphere programme/international global atmospheric chemistry SAFARY. 92 field experiment: background and overview. *Jour. Geographical Research*, **101**(D19):23,521–23,530.

[10] Knutsson, H. and Westin, C.-F., (1993). Normalized and differential convolution. In *Proceedings CVPR'93, New York City, NY*, pp. 515–523, IEEE. Washington, DC: IEEE Computer Society Press.

[11] Deriche, R., (1990). Fast algorithms for low-level vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **12**(1):78–87.

[12] Marquard, L., (1998). *Modellierung des Strahlungstransports in der Erdatmosphäre für absorptionsspektroskopische Messungen im ultravioletten und sichtbaren Spektralbereich.* Dissertation, University of Heidelberg.

[13] Foley, J., van Dam, A., Feiner, S., and Hughes, J., (1990). *Computer Graphics: Principles and Practice, Second Edition.* Addison-Wesley Systems Programming Series. Reading, MA: Addison-Wesley.

38 Tracking "Fuzzy" Storms in Doppler Radar Images

John L. Barron[1], Robert E. Mercer[1], David Cheng[1], and Paul Joe[2]

[1]The University of Western Ontario, London, Ontario
[2]Atmospheric Environmental Services, Toronto, Ontario

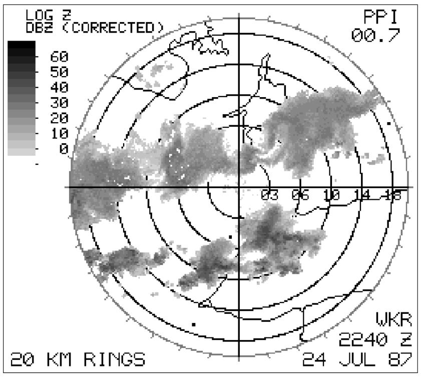
38.1 Introduction

Because of the devastation caused by severe storms, the forecasting of severe storm movement is one of the most important tasks facing meteorologists. To help with this task we have developed an automatic *storm-tracking system*. Tracking past storm movement is a prerequisite to forecasting future storm movement and minimizing property damage.

Physically, a *storm* is an area of updraft and downdraft turbulence of air and water particles. In the updraft, a rain cloud is formed from the collision of water particles. When the water particles are too large

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

a



b

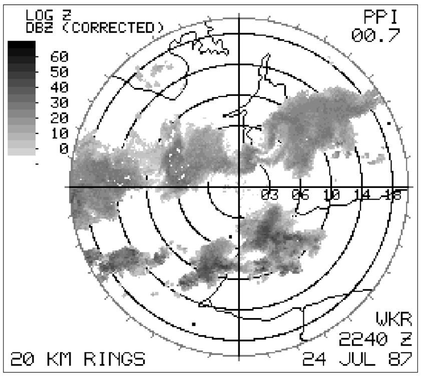


Figure 38.1: *Doppler radar:* ***a*** *density; and* ***b*** *radial velocity image.*

to be sustained by the updraft, they fall as precipitation in the form of snow, rain or hail. The lifecycle of an average storm containing a single storm cell usually ranges from 10 to 20 min. On the other hand, severe storms having multiple storm cells tend to have a much longer lifespan of up to a few hours and therefore have the potential to cause significant damage.

38.2 Problems of storm tracking

Since 1985, the Cloud Physics Research Division of the *Atmospheric Environment Service* (AES) of Canada has been developing a Doppler radar system to detect severe storms such as thunderstorms and *tornadoes*. The *Doppler radar* generates intensity and radial velocity images, examples of which are shown in Fig. 38.1.

These images are preprocessed to remove pixels corresponding to lines, bars and writing. Intensity interpolation is used to represent the values of removed pixels. Typically, there are 0 to 20 potential severe storms per image.

The recognition and tracking of storms in these radar images is currently performed manually by human experts and the task is time consuming. To improve the efficiency and quality of weather forecasting, AES is interested in developing an automatic storm-tracking system for use in their operations. Towards this end, we have developed a tracking program with visualization capabilities that uses a *hypothesize and verify* model to detect storms in radar images and construct storm tracks. We first hypothesize storm masses in the Doppler radar intensity images. Then we verify the correctness of these hypothesized storms by tracking them over time. If an hypothesized storm can be tracked over a desired number of frames, we conclude that the storm is a valid storm and we record its track. When all potential storms are verified, a set of valid storm tracks will be taken as outputs.

38.2.1 Aliasing of data

The Doppler radar intensity image sequences that we have used as experimental data are actual data obtained from the AES radar station at King City, Ontario. Because of the operational nature of the radar at the King City radar station, an intensity/velocity image is generated every 10 min. During the 10-min interval, storms can appear or disappear. Moreover, other storm properties such as size, intensity and shape can change significantly. Under such a low sampling rate, the data can easily be *aliased*. If the matching of storms in image sequences uses storm properties as a measure of correspondence or if the storms are not allowed to remain unmatched, this data aliasing can directly affect the accuracy and performance of the storm-tracking system.

38.2.2 The merging and splitting of storms

Through studying the storm movements in several Doppler intensity image sequences, two important phenomena are observed: a single storm may *split* into several separate storms; and two or more storms may *merge* into one storm in the next image. We have developed the concept of *pseudo-storms* to accommodate this behavior.

38.3 Background: Zhang/Krezeski's algorithms

A complete literature survey is available in Cheng et al. [1]. In the following, we describe only our previous work.

The early work done on automatic storm tracking can be found in a Master's thesis by Zhang [2] and in a paper by Krezeski et al. [3]. Here we highlight a few features in their work, as they form the basis of our current work.

Based on a *merge-and-split algorithm* developed by Horowitz and Pavlidis [4], Zhang [2] devised a similar *region-growing algorithm* to identify storm masses in a sequence of Doppler radar intensity images. An image is post-processed into a square digital image that is initially divided into a user-defined number of square regions or subnodes. A subnode $S = \{p_1, p_2, \ldots, p_n\}$ can be considered as a set of n pixels, where each pixel p_j represents a gray intensity level. Subnodes are then merged or split according to rules associated with a global threshold $T_{\bar{z}}$:

- A subnode S is split into four smaller, equal-sized square subnodes if the difference between the maximum and the minimum pixel values of the subnode is greater than the threshold $T_{\bar{z}}$

$$\max(S) - \min(S) > T_{\bar{z}} \tag{38.1}$$

- Four adjacent subnodes S_1, S_2, S_3 and S_4 are merged into a single subnode if the difference between the maximum and the minimum pixel values of the four subnodes is less than the threshold $T_{\bar{z}}$

$$\max\left(\bigcup_{j=1}^{4} S_j\right) - \min\left(\bigcup_{j=1}^{4} S_j\right) \leq T_{\bar{z}} \tag{38.2}$$

The merge-and-split operations are mutually exclusive. That is, a subnode that is split cannot be merged and vice versa. When no subnode can be further split or merged, the original image is completely segmented into many different square subregions that are represented in a pyramid data structure. Then a grouping operation is performed to group adjacent regions to form an irregularly shaped region. A region whose size and average intensity are above two other thresholds is hypothesized as a storm. The center of mass of an hypothesized storm region is then considered as an hypothesized storm center.

By modifying Barnard and Thompson’s spatial relaxation-labeling algorithm [5], Zhang has developed a temporal relaxation algorithm to track storm centers represented by normal Euclidean points over time. The algorithm is based on the *smoothness assumption* of storm movement. Initially, disparities between the storm centers in adjacent images are constructed and each disparity is associated with a certainty value. The certainty value is then modified by relaxation on distance and angle compatibility. In the end, disparities with the highest certainty values are chosen as the best matches.

To handle the merging and splitting of storms, Zhang’s algorithm allows single storms to be matched to several storms in the immediately previous or following images. This approach was less successful than desired because her tracking algorithm could incorrectly match premerged single small storms to a larger single storm.

Krezeski et al. [3] added *property coherence* and the concept of pseudo-storms to improve Zhang’s tracking algorithm. Their algorithm yielded some promising experimental results. Property coherence [6] allows multiple features of a storm to be tracked over time in addition to the location of the storm center. Five additional storm properties they considered were average intensity, storm size, velocity variance, storm shape and orientation, and convexity. The *velocity variance* property is the only use made of the Doppler velocity in our work to date. Areas of high-velocity variance corresponded to storms.

Krezeski et al. [3] use the same *merge-and-split algorithm* as employed by Zhang [2] to segment a radar intensity image and to hypothesize storm masses. Minor modifications are made to the algorithm to extract additional storm properties. A disparity is constructed from a storm in one image to a storm in the other if the distance between their storm centers is within a certain threshold T_d. Connection between two adjacent disparities is established if the angle between the two is less than another threshold T_θ. A certainty value is associated with each disparity and is determined by a set of compatibility values. Initially, the compatibility between adjacent disparities is given by

$$C_0(d_1,d_2) = w_6 c_d(d_1,d_2) + w_7 c_\theta(d_1,d_2) + \sum_{k=1}^{5} w_k \left(\frac{f_k(d_1) + f_k(d_2)}{2} \right) \tag{38.3}$$

where $w_1, w_2, \dots, w_7$ are normalized weights that sum to 1, c_d is the length compatibility function, c_θ is the angle compatibility function, and f_k is the storm property function for property p_k, where $k = 1, 2, \dots, 5$. For each property p_k, the property function computes the compatibility value of that property for each disparity d corresponding to two temporally connected storms s_1 and s_2 in adjacent images

$$f_k(s_1,s_2) = 1 - \left(\frac{\mid p_k(s_1) - p_k(s_2) \mid}{\max(\, p_k(s_1), p_k(s_2)\,)} \frac{M_k}{M_k - m_k} \right) \tag{38.4}$$

where M_k and m_k denote, respectively, the maximum and the minimum values of property p_k of all storms s. The certainty value of each disparity is later refined iteratively by relaxation.

To handle situations where storms merge and split, Krezeski introduces the notion of a *pseudo-storm*. His idea is based on the work by Einfalt et al. [7]. Einfalt suggested that storms that are close together could be considered as a single entity for comparison with other storms in adjacent images. Although Krezeski's algorithm with pseudo-storm is still under development, initial experiments show that the merging and splitting of storms can be handled very well by matching real storms to pseudo-storms in adjacent images [3].

Krezeski's result showed some problems that can be attributed to the deformability and changing intensity values in the storm. The representation of the location of a storm in an image (which is required by the relaxation algorithm) is a point. The method used to determine this point (center of mass calculation) is sensitive to the properties of the storm (deformability, changing intensities). Although the storm moves in one direction, the algorithm can actually record a movement in the opposite direction due solely to the somewhat arbitrary location of the center of mass of the storm in the subsequent image. This undesirable movement of the point representing the storm affects the performance of the relaxation algorithm. To obtain realistic results with the methodology described in Krezeski et al. we had to set thresholds in the relaxation algorithm at levels that sometimes allowed incorrect tracks to be generated.

38.4 Fuzzy storm centers

To diminish the effect of the arbitrariness of storm location, we have modified the representation of the center of a storm from a Euclidean point to a *fuzzy point* [1, 8]. (For a review of the concepts of fuzzy image processing see Volume 2, Chapter 22.) A fuzzy point is a circle whose inner region represents the uncertainty of the location of a targeted point (a more complete description of our fuzzy storm geometry is available in Mercer et al. [9]). Because the fuzzy point represents the uncertain location of the storm center, we refer to the fuzzy point as the fuzzy storm center. Our current work [8, 10] uses fuzzy points, *fuzzy vectors*, fuzzy lengths of fuzzy vectors, and *fuzzy angles* between two nonzero fuzzy vectors in the relaxation framework to produce more realistic storm tracks. Note that our interpretation of a fuzzy point is not based on the theory of fuzzy sets [11]; rather, it is in the spirit of *fuzzy geometry* ([12, 13], and Volume 2, Section 22.5.1).

38.4.1 Fuzzy point algebra

A *fuzzy point* $P = \langle c, r \rangle$ is a circle with center $c = (c_x, c_y)$ and radius r. It represents a region where a targeted point can arbitrarily be located. We denote the set of all fuzzy points as $\mathbf{P}$. We define the distance δ between two fuzzy points $P_1 = \langle c_1, r_1 \rangle$ and $P_2 = \langle c_2, r_2 \rangle$ as

$$\delta(P_1, P_2) = \|c_1 - c_2\|_2 + |r_1 - r_2| \tag{38.5}$$

where $(\mathbf{P}, \delta)$ forms a metric space. If we consider P as a subset of $\mathbf{R}^2$, then a *fuzzy vector* $\overrightarrow{P_1P_2}$ from fuzzy point P_1 to fuzzy point P_2 is the set of all displacement vectors from a point in P_1 to a point in P_2. The *fuzzy length* of this fuzzy vector is then defined as the set of lengths of all displacement vectors in $\overrightarrow{P_1P_2}$. This set can be represented as the real interval $[d_{\min}, d_{\max}]$, where $d_{\min}$ and $d_{\max}$ are, respectively, the least and greatest distance between any two points in the circles representing the fuzzy points P_1 and P_2. Any fuzzy vector with fuzzy length $[0, d]$, $d \geq 0$, is considered as a zero fuzzy vector. The *fuzzy angle* subtended by a nonzero fuzzy vector $\overrightarrow{P_2P_3}$ relative to a nonzero fuzzy vector $\overrightarrow{P_1P_2}$ is defined as the set of angles subtended by any displacement vector in $\overrightarrow{P_2P_3}$ relative to a displacement vector in $\overrightarrow{P_1P_2}$ having touching heads and tails, respectively. The angle between the two displacement vectors θ can be determined by the dot-product. Similar to the fuzzy length of a fuzzy vector, the fuzzy angle between two nonzero fuzzy vectors is a real interval $[\theta_{\min}, \theta_{\max}] \subseteq [0, \pi]$. However, determining the two endpoints $\theta_{\min}$ and $\theta_{\max}$ is not trivial. Currently, we use an $O(n)$ search algorithm to approximate the two endpoints.

38.4.2 Storm hypothesis

A region-splitting algorithm with dynamic thresholding is used to determine storm masses in a radar intensity image [8, 10]. This algorithm replaces the one in Krezeski et al. [3] that uses Horowitz and Pavlidis' merge-split algorithm [4]. The merge-split process in Krezeski et al. [3] is determined by a threshold on the difference between the maximum and minimum intensity levels of adjacent regions. We have observed that this criterion is sensitive to outliers in the data. It also requires the appropriate threshold to be chosen manually for each image. In our new algorithm, we use the standard deviation s of the average intensity level of a region to govern the splitting process. We find that thresholding using s is very robust and no user interaction is required.

A square image S is initially at level 1 of a quad-tree and is divided into four equally sized subnodes S_1, S_2, S_3 and S_4 that correspond to the north-western, north-eastern, south-eastern and south-western regions, respectively.

For each subnode S_n, where $n \in \Lambda^+$ and $\Lambda = \{1, 2, 3, 4\}$, we compute the mean $\bar{z}$ and the standard deviation s_z of the intensity levels of all pixels in S_n. The splitting criterion of the subnode is based on the value of s_z. The value of s_z indicates how well the value of $\bar{z}$ represents the average intensity level of all pixels in S_n. We currently set a threshold T_s on s_z to 10.

- If s_z is large, as indicated by $s_z \geq T_s$, then the intensity levels of most pixels in S_n differ significantly from $\bar{z}$ and therefore the subnode S_n is split into four smaller subnodes S_{n1}, S_{n2}, S_{n3} and S_{n4} at the next level.
- On the other hand, if s_z is small, as indicated by $s_z < T_s$, then we consider $\bar{z}$ to be representative of the intensity levels of most pixels in S_n and we compare the value of $\bar{z}$ to the dynamic threshold $T_{\bar{z}}$; $T_{\bar{z}}$ is based on the mean $\bar{z}'$ and standard deviation s'_z of the intensity levels of a subset of pixels in the image with intensity levels greater than $z_{\min} = 16$. It is computed as $T_{\bar{z}} = \bar{z}' + ks'_z$, where $k = -0.5$. If we have $\bar{z} \geq T_{\bar{z}}$, then the subnode S_n will be considered as a part of a potential storm and will be marked for further processing.

The foregoing splitting process continues recursively until no subnode can be split further. Neighboring subnodes that are marked for further processing will be grouped into a single region

$$R = S_{k_1} \cup S_{k_2} \cup \cdots \cup S_{k_n} \tag{38.6}$$

if they are connected. We say that a subnode S' is *connected* to a subnode S with a distance d if all of the following conditions hold:

$$\begin{cases} l_y + d & \geq & u'_y \\ u_y - d & \leq & l'_y \\ l_x + d & \geq & u'_x \\ u_x - d & \leq & l'_x \end{cases} \tag{38.7}$$

where (u_x, u_y) and (l_x, l_y) denote the upper-left and lower-right corners of S, respectively; (u'_x, u'_y) and (l'_x, l'_y) denote the upper-left and lower-right corners of S', respectively; and d is a threshold currently set to 2 pixels.

38.4.3 Construction of fuzzy storm centers

After region splitting we have a set of regions $\{R_1, R_2, \ldots, R_n\}$ in the Doppler radar intensity image that are hypothesized as storms. To represent the location of each storm region R_j using a fuzzy point $P = \langle c, r\rangle$, we first compute the weighted averages $\bar{x}$ and $\bar{y}$ of the centers of all subnodes forming the region in the x- and y-direction, respectively; and also the corresponding standard deviations s_x and s_y. Then the center of the fuzzy point P is taken as $c = (\bar{x}, \bar{y})$; and the radius of the fuzzy point is determined by $r = k_r \max(s_x, s_y)$, where k_r is a parameter to control the size of the fuzzy point. We currently use $k_r = 0.5$. We interpret the foregoing construction as fitting a quasi-circular Gaussian surface through the storm region. Because the data points in the storm region do not necessarily spread around the peak of a Gaussian surface, we could not employ a least-square method to perform a Gaussian fit.

38.5 Incremental relaxation-labeling algorithm

Once a set of storms has been hypothesized for the radar intensity image sequence, the correctness of these storms can be verified by tracking them over time. Our tracking algorithm is based on Krezeski’s temporal relaxation algorithm with property coherence [3]. We have selected the size of a fuzzy storm center as a property to be coherent over time.

Let S_k be an hypothesized fuzzy storm center in the kth image. A disparity represented by a fuzzy vector $\overrightarrow{S_j S_{j+1}}$ is constructed from S_j to S_{j+1} if the *infimum* of the fuzzy length of the fuzzy vector is less than a threshold T_d, which is set to a default value of 10 pixels; and concurrently, the two fuzzy storm centers have compatible sizes. We define a property function f_s to measure the *size-compatibility* of two fuzzy storm centers $S_1 = \langle c_1, r_1 \rangle$ and $S_2 = \langle c_2, r_2 \rangle$ as

$$f_s(S_1, S_2) = \begin{cases} 1 - \dfrac{|r_1 - r_2|}{\max(r_1, r_2)} & \text{if } r_1 > 0 \text{ or } r_2 > 0, \\ 1 & \text{otherwise} \end{cases} \tag{38.8}$$

A size-compatibility threshold T_{sc} is set to 0.5. Note that if T_{sc} is set to 1, then a disparity will be constructed between two fuzzy storm centers only when they have exactly the same size. On the other hand, if T_{sc} is set to 0, then the size-compatibility criterion is effectively removed. We measure the partial compatibility between two adjacent disparities using a weighted sum of three components: *length compatibility C_d*; *angle compatibility C_θ*; and *size compatibility C_s*. The overall compatibility function is defined as

$$C = w_d C_d + w_\theta C_\theta + w_s C_s \tag{38.9}$$

where w_d, w_θ and w_s are normalized weights such that $w_d + w_\theta + w_s = 1$. We currently use $w_d = 0.2, w_\theta = 0.2$ and $w_s = 0.6$. Values for these and other weight coefficients (in the following) were chosen by empirical observation. Two adjacent disparities are connected together if their compatibility value is greater than a threshold

$$C\left(\overrightarrow{S_j S_{j+1}}, \overrightarrow{S_{j+1} S_{j+2}}\right) > T_c \tag{38.10}$$

where T_c is currently 0.2. When all qualified adjacent disparities have been linked together, the certainty of each disparity is refined iteratively by relaxation on the overall compatibility among its adjacent disparities. Consider a disparity $d = \overrightarrow{S_j S_{j+1}}$. The initial certainty of the disparity, denoted as $p_0(d)$, is set to $f_s(S_j, S_{j+1})$.

During each iteration, we apply both spatial and temporal consistency constraints to compute the supporting and contradictory evidence of the disparity using the compatibility values. Let E_s and E_c denote the supporting and contradictory evidence, respectively. Let n_s and n_c denote the number of supporting disparities and the number of contradictory disparities, respectively. These four quantities are reset at the start of each iteration.

- To apply the temporal consistency constraint, for each adjacent disparity d_t to d of the form $d_t = \overrightarrow{S_{j-1} S_j}$ or $d_t = \overrightarrow{S_{j+1} S_{j+2}}$, we compute the compatibility at the kth iteration ($k > 0$) between the two disparities as

$$C_k(d, d_t) = w_1 C(d, d_t) + w_2 \left(\frac{p_{k-1}(d) + p_{k-1}(d_t)}{2} \right) \qquad (38.11)$$

where w_1 and w_2 are normalized weights that sum to 1. We currently use $w_1 = 0.4$ and $w_2 = 0.6$. If $C_k(d, d_t) > T_k$ (we use $T_k = 0.6$), then we add $p_{k-1}(d)$ to E_s and increase n_s by 1. Otherwise, we add $p_{k-1}(d)$ to E_c and increase n_c by 1.

- To apply the spatial consistency constraint, for each disparity d_s that has the same head storm or tail storm as d, if $p_{k-1}(d) \geq p_{k-1}(d_s)$, then we add $p_{k-1}(d)$ to E_s and increase n_s by 1. Otherwise, we add $p_{k-1}(d)$ to E_c and increase n_c by 1.
- The certainty of the disparity d at the kth iteration is modified by

$$p_k(d) = \begin{cases} \frac{1}{2}\left(1 + \frac{w_s E_s - w_c E_c}{w_s E_s + w_c E_c}\right) & \text{if } E_s \neq 0 \text{ or } E_c \neq 0, \\ 0 & \text{otherwise} \end{cases} \qquad (38.12)$$

where w_s is the weight of the supporting evidence and is computed as $w_s = n_s/(n_s + n_c)$, and w_c is the weight of the contradictory evidence and is computed as $w_c = n_c/(n_s + n_c)$.
- The iterative process stops at the kth iteration when the certainty of each disparity has converged to the desired level of confidence (ε), say to n decimal places (we use $n = 6$)

$$\varepsilon = \left| p_k(d) - p_{k-1}(d) \right| < 10^{-n} \qquad (38.13)$$

for each disparity d or the maximum number of iterations has been reached: $k \to T_k$, where T_k is currently set to 20.

Once the relaxation process has converged, we construct a set of all longest tracks such that each disparity has a final certainty over a threshold T_p (we use $T_p = 0.85$). We choose a subset of these tracks, with the condition that no storm lies upon more than one chosen track.

We have changed the implementation of the algorithm from processing of a complete image sequence of known length (i. e., *batch mode*) to be in *incremental mode*. Given n images the hypothesized storm disparities are relaxed. When an $(n+1)$th image is added, the relaxation is restarted using the results for the first n images plus the hypothesized storms of the $(n + 1)$th image. We have observed empirically that the result is always the same as if all $n+1$ images had been initially relaxed. The difference in computation speed between the two methods is insignificant as relaxation converges within 10 iterations in either mode. The *incremental algorithm* allows us to view the current storm tracks as the data become available.

38.6 An X-Window-based storm-visualization program

We have developed and implemented a storm-visualization program using the standard X11 Athena toolkit. We have chosen the Athena toolkit for its portability across different UNIX platforms. Our program has been compiled and tested on several platforms including a Sun Sparcstation, an IBM RS/6000 workstation, and an Intel Pentium workstation and notebook. The main design goal of this storm-visualization program was to allow the study of storm movements in radar intensity image sequences and to visualize fuzzy storm centers and storm tracks generated by our storm-tracking algorithms. The interested reader can ftp to `ftp.csd.uwo.ca` (login: your email address and password: anonymous) and cd to pub/vision/DCHENG to obtain the compete data set and C/X windows program. Also, the programs/data sets are included in the CD-ROM for this handbook (`/software/38`).

The original algorithms by Zhang and Krezeski used polylines to draw tracks connecting storm centers represented by Euclidean points. We find this method inappropriate for fuzzy storm centers because the actual location of the storm center can be anywhere inside the circumference of a fuzzy point. In addition, the zig-zag look of a polyline fails to represent the smooth movement of a storm. Therefore, we have used cubic B-spline approximation [14] with storm center averaging to draw the storm tracks. We did not use Cubic B spline *interpolation* because this would involve solving a set of simultaneous linear equations and we do not need the curve to pass through the center of a fuzzy point. Because we select the centers of the fuzzy points as the knot points of the spline curve, the curves have many bends. To improve the smoothness of the spline curve, we used a simple smoothing filter to apply on the spline knots. For a set of spline knots P_{-1} to P_{n+1}, we construct a set of *smoothed* spline knots P'_{-1} to P'_{n+1} using a smoothing filter F: $P'_j = F(P_j) = 1/3(P_{j-1} + P_j + P_{j+1})$, if $j = 1, 2, \ldots, n-1$, and P_j otherwise. We then use this set of smoothed spline knots P'_j to plot the spline curve.

The interaction between the *storm visualization* program and a user is performed through a pop-up menu (see Fig. 38.2 for a screen-shot of the program with the pop-up menu activated).

38.7 Experimental results

A complete set of experimental results can be found in Cheng et al. [1]. Due to space limitations we present only the results for the series-18 sequence made from the radar station at King City, Ontario.

In our algorithm there are a number of thresholds to be set. For the region-splitting algorithm, z_{min} governs the background pixel in-

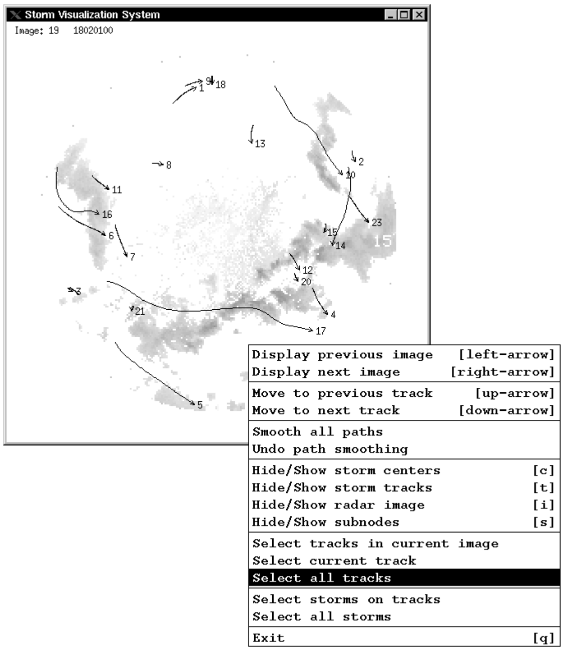

Storm Visualization System
Image: 19 18020100
9 18
1
13
8
2
10
11
16
23
6
15
14
7
12
20
3
21
4
17
5
Display previous image [left-arrow]
Display next image [right-arrow]
Move to previous track [up-arrow]
Move to next track [down-arrow]
Smooth all paths
Undo path smoothing
Hide/Show storm centers [c]
Hide/Show storm tracks [t]
Hide/Show radar image [i]
Hide/Show subnodes [s]
Select tracks in current image
Select current track
Select all tracks
Select storms on tracks
Select all storms
Exit [q]

Figure 38.2: *A screen-shot of our X-window-based storm-visualization program with the pop-up menu activated.*

tensity, K governs the dynamic threshold and T_s governs the region-splitting operation. For all results in this paper we use the values $z_{min} = 16$, $K = -0.5$ and $T_s = 10.0$. In the tracking algorithm the construction of a disparity between two fuzzy storm centers in adjacent images is controlled by two thresholds: T_d governs the closest distance between two fuzzy storms and T_{sc} governs the size compatibility between two fuzzy storms. Usually, T_s is 10, but occasionally it is tightened to 5 to remove unwanted storms; T_{sc} is usually set at 0.5. As we noted in the foregoing, $T_{sc} = 0.0$ turns off size compatibility while $T_{sc} = 1.0$ enforces strict size compatibility.

This sequence has 30 images consisting of complex storm movements (it represents our most complex storm sequence). Storms are moving from the northeast to the southeast and from west to east. In the end both storm movements merge into one big storm moving southeast. This can be seen from the verified storm tracks shown in Fig. 38.3, which show the verified storms tracks for storms in images 5, 12, 19 and 30. Other storm sequences are available via our ftp site or the CD-ROM for this handbook.

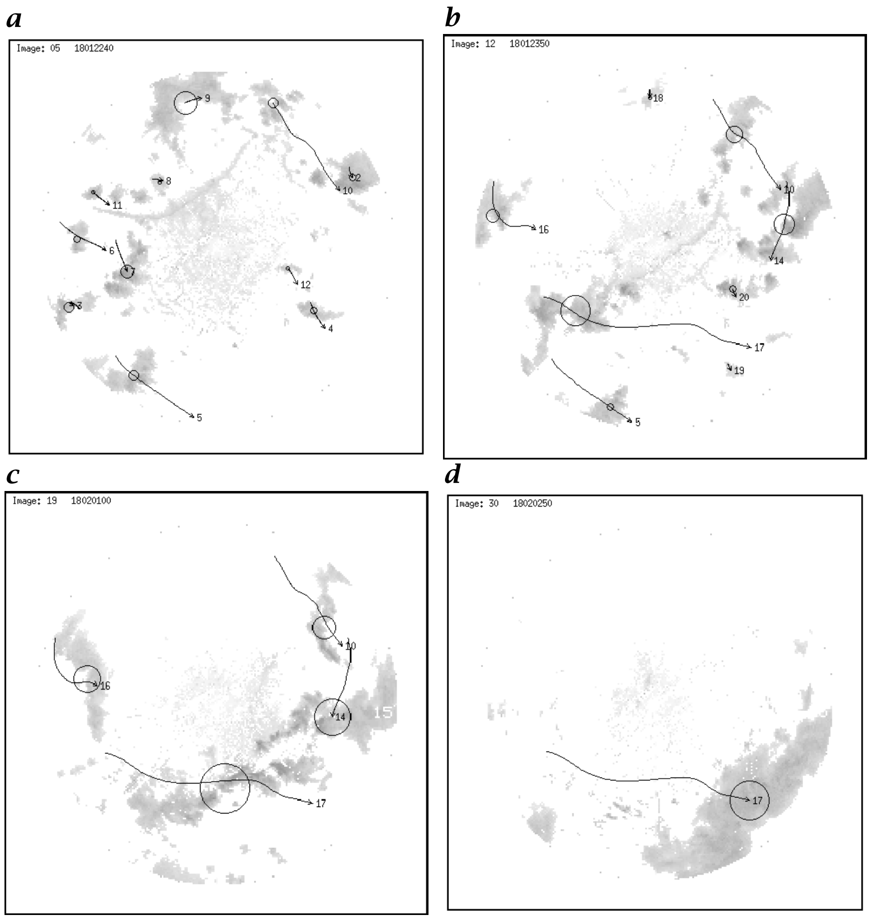


Figure 38.3: *The storm tracks for the* ***a*** 5th, ***b*** 12nd, ***c*** 19th *and* ***d*** 30th *images of the 18-series;* $T_d = 5.0$ *and* $T_{sc} = 0.6$.

38.8 Conclusions

By using fuzzy storm centers and relaxation labeling we are able to obtain storm tracks that are long and smooth and which closely match human perception of a "motion picture" of a storm image sequence. Future plans include testing our algorithm on other types of tracking problems, for example, tracking clouds in satellite imagery or aerial views of oil/chemical spills in lakes and rivers. We also plan on extending this algorithm to 3-D. Slices of Doppler image data at different radial angles (heights) are now available and it should be possible to hypothesize and track 3-D fuzzy storms rather than 2D fuzzy storms as we do now. We believe that Doppler velocity data will be more useful in this extension.

38.9 References

[1] Cheng, D., Mercer, R., Barron, J., and P.Joe, (1998). Tracking severe weather storms in Doppler radar images. *Intl. Journal of Imaging Systems and Technology (to appear).*

[2] Zhang, H., (1991). *Storm detection in radar images.* Master's thesis, University of Western Ontario, Dept. of Computer Science.

[3] Krezeski, D., Mercer, R. E., Barron, J. L., Joe, P., and Zhang, H., (1994). Storm tracking in Doppler radar images. In *Proc. IEEE International Conf. on Image Processing (ICIP94), Austin, Texas*, Vol. III, pp. 226–230.

[4] Horowitz, S. L. and Pavlidis, T., (1976). Picture segmentation by a tree traversal algorithm. *Journal of the ACM*, **23**(2):368–388.

[5] Barnard, S. T. and Thompson, W. B., (1980). Disparity analysis of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **2**(4):333–340.

[6] Sethi, I. K., Chung, Y. K., and Yoo, J. H., (1992). Correspondence using property coherence. *SPIE Applications of Artificial Intelligence X: Machine Vision and Robotics*, **1708**:653–662.

[7] Einfalt, T., Denoeux, T., and Jacquet, G., (1990). A radar rainfall forecasting method designed for hydrological purposes. *Jour. Hydrology,* **114(3-4)**:229–244.

[8] Cheng, D., Mercer, R. E., Barron, J. L., and Joe, P., (1996). Tracking fuzzy storm centers in Doppler radar images. In *Proc. IEEE International Conf. on Image Processing (ICIP96), Lausanne, Switzerland*, Vol. II, pp. 959–962.

[9] Mercer, R., Barron, J., and Cheng, D., (1998). Fuzzy points: Algebra and application. *Submitted.*

[10] Cheng, D., (1996). *Tracking fuzzy storm centers in Doppler radar images.* Master's thesis, University of Western Ontario, Dept. of Computer Science.

[11] Zadeh, L. A., (1965). Fuzzy sets. *Information and Control*, **8**:338–353.

[12] Rosenfeld, A., (1992). Fuzzy geometry: An overview. In *Proc. First IEEE Conf. Fuzzy Systems, San Diego, CA*, pp. 113–117.

[13] Rosenfeld, A., (1994). Fuzzy plane geometry: Triangles. *Pattern Recognition Letters*, **15**(12):1261–1264.

[14] Burger, P. and Gillies, D., (1989). *Interactive Computer Graphics.* New York: Addison Wesley.

39 Detection of Dendritic Spine Synapses in Confocal Microscope Images

Rolf Watzel[1], Wolfgang Hilberg[1], Henning Scheich[2], and Katharina Braun[2]

[1]Elektrotechnik, Technische Universität Darmstadt, Germany
[2]Leibniz Institut für Neurobiologie, Magdeburg, Germany

39.1 Introduction

This chapter reports an application of 3-D image processing techniques in the field of neuromorphology. The objective is to count and describe *dendritic spine synapses*. These are the links that establish signal flow from one *neuron's axon* to another neuron's *dendrite* as illustrated in Fig. 39.1. It has been observed that the density [1, 2, 3] and also the shape and size of spine synapses are altered during learning events and as a consequence of aging or mental diseases [4, 5]. This leads to the conjecture that the morphology of these synapses may be correlated with learning processes. Therefore, an image analysis tool has been designed to perform analysis of the spine distribution in 3-D images recorded by a *confocal laser scanning microscope*, [6, 7].

Confocal microscopy has been chosen because it is capable of covering a section of dendrite of considerable length at nearly sufficient

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

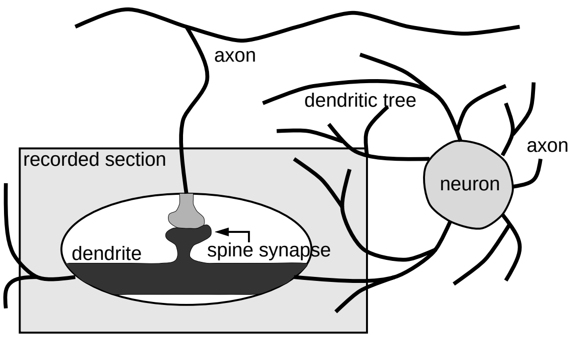


Figure 39.1: *Schematic example of a spine synapse and its environment.*

resolution at synaptic level. Electron microscopy, in contrast, would give too much data to examine a large number of synapses. The choice of confocal microscopy imposes physically unavoidable image degradation (Section 39.2) that imposes problems in image interpretation at points where spines meet closely. In such cases two spines may be regarded as a single furcated one. It is also possible that two spine necks meeting closely may generate a light distribution similar to that of a head, and adjacent heads may appear as a single one. On the other hand, thin spine necks may be too weak to be recognized, and thus it can not be determined where the spine is attached to a dendrite. All this gives rise to our efforts to reconstruct the dye distribution in the best possible way with respect to preservation of *topology* and morphological features of the tissue.

To achieve this, our approach makes use of the following topological properties of the arrangement of dendrites and spines: first, they are simply connected. This means that they comprise a connected region in space and do not include any cavities or “tunnels.” A spine comprises a neck or a head or both. It is connected to a dendrite at exactly one point. Spines are not mutually connected.

Apart from a topological description, they can be described by their size. Typically, spine length is $<3\,\mu$m. The neck diameter may range down to 80 nm. Dendrites are typically longer than 8 μm. Their diameter is on the order of 1 μm.

The recognition process implemented here includes multiple stages starting with image restoration and enhancement. The image is segmented into foreground and background by thresholding and transformed to a graph via a *medial axis transform*. Higher-level shape analysis and recognition is performed on this graph. All examples given in the following sections refer to the image depicted in Fig. 39.2. The area covers $21.8 \times 15.6 \times 11.7\,\mu$m. It has been cut out from a recorded vol-

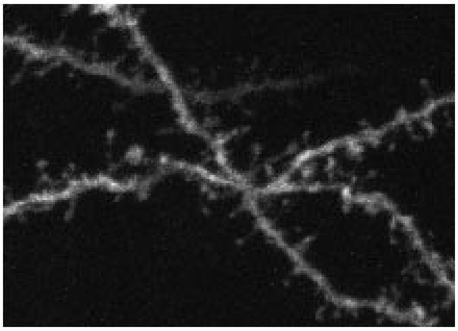


Figure 39.2: *The example image of this contribution in the xy-plane.*

ume of $51.2 \times 51.2 \times 11.7\,\mu$m. All original images and various stages of the processed images are also contained on the CD-ROM `(/images/39)`.

39.2 Data acquisition by confocal microscopy

Data acquisition includes preparation of histological slices, staining neurons of interest with a dye (*Lucifer Yellow*), and recording the specimen by a Leica TCS^{4D} confocal laser scanning microscope (CLSM). Lucifer Yellow is iontophoretically injected into a neuron and will diffuse into the dendritic tree and into the spine synapses. It will neither pass the membrane nor penetrate into the *presynapses*. In this application it is assumed that diffusion has reached equilibrium such that the dye density is at least locally constant. The volume included in one scan is typically sized $51.2 \times 51.2 \times 10\,\mu$m. This corresponds well to the typical length of dendrite branches to be observed ($\approx 10 - 50\,\mu$m). Voxel size is $0.1 \times 0.1 \times 0.1\,\mu$m. Small synaptic structures (the neck diameter of a small spine may be 80 nm) are thus undersampled, but the optical transfer function (OTF) of the CLSM provides a natural *antialiasing filter* and *deconvolution* up to the resolution of synaptic structures seems to be impossible. The lateral full half-width of the *point-spread function* (PSF) is approximately $0.5\,\mu$m and its axial full half-width is approximately $1.4\,\mu$m. A $100\times$ lens with NA 1.4 and zoom factor 2 has been used; laser wavelength was 430 nm and the emission wavelength of Lucifer Yellow is 490 nm.

The choice of voxel size and image dimensions is a trade-off between *sampling rate* on one side and memory requirements, scanning time and computation time on the other. The smallest scale is also limited by the *optical resolution* that can be achieved with the given equipment. Gray values are encoded as 8 bit integers.

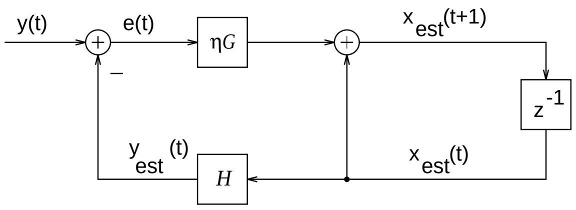


Figure 39.3: *Structure of the deconvolution filter.*

39.3 Image restoration

The observation through the CLSM comprises a mapping from the dye distribution to a digital image that is degraded by *blurring* and noise (Volume 1, Chapter 21 and Volume 2, Chapter 5). Due to the presence of random noise, *restoration* can not be achieved perfectly, but significant improvements are possible. Here, a combination of noise reduction by smoothing (Volume 2, Chapter 7) and *deconvolution* (see also Volume 1, Chapter 21) is used. For simplicity and reduction of computational effort, the PSF is regarded as a scaled *Gaussian function*.

The smoothing filter has been implemented as the averaging filter $^3\mathcal{R}\,^5\mathcal{R}\,\mathcal{B}^2$ (Volume 2, Chapter 7). It approximates a Gaussian filter with $\sigma = 1.8$. Due to the separability of the Gaussian function, σ simply adds to the width of the assumed Gaussian PSF (Volume 1, Chapter 21).

We found that the *Jansson-van Cittert method* described by Frieden [8] and Conchello and Hansen [9] gives best results among techniques such as *van Cittert deconvolution*, Richardson Lucy (White [10]) and gradient descent update. A detailed comparison of these techniques would be beyond the scope of this chapter and is omitted here.

The Jansson-van Cittert method falls into a class of dynamic filters as illustrated in Fig. 39.3; $\boldsymbol{y}$ denotes the input image and $\boldsymbol{x}_{est}(t)$ the estimated solution of the *inverse problem*. The *convolution* operator $\mathcal{H}$ performs convolution with the PSF. If images are regarded as vectors of dimension N, the convolution operator $\mathcal{H}$ becomes an $N \times N$ matrix with a shifted version of the reverse PSF in each column. The state image is updated by applying an operator $\mathcal{G}$ to the error image $\boldsymbol{e}(t)$ and adding the result to $\boldsymbol{x}_{est}(t)$

$$\boldsymbol{x}_{est}(t+1) = \boldsymbol{x}_{est}(t) + \eta\boldsymbol{G}\underbrace{(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}_{est}(t))}_{\boldsymbol{e}(t)} \tag{39.1}$$

$$\boldsymbol{x}_{est}(1) = \boldsymbol{y} \tag{39.2}$$

If $\mathcal{G}$ is unity, the filter implements the van Cittert method, and in case of $\mathcal{G} = \mathcal{H}^T$ it will establish a *gradient descent method*. Insertion of

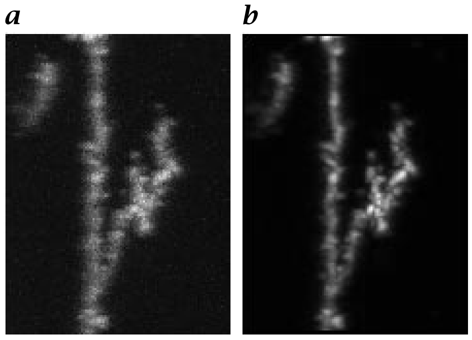


***Figure 39.4:** **a** Original; and **b** deconvolved image in the yz plane.*

proper log and exp operators will transform the gradient method to *Richardson Lucy deconvolution*. The Jansson-van Cittert method uses a nonlinear operator $\mathcal{G}$ according to

$$\boldsymbol{x}_{est}(t+1) = \boldsymbol{x}_{est}(t) + \eta r(x)\boldsymbol{e}(t) \tag{39.3}$$

with the pointwise factor

$$r(x) = 255 - 2|128 - x| \tag{39.4}$$

It constrains state values x to $0 \le x \le 255$. In each step we compute η to minimize the mean squared error

$$\eta(t) = \frac{\boldsymbol{e}(t)^T(\boldsymbol{H}r(x)\boldsymbol{e}(t))}{||\boldsymbol{H}r(x)\boldsymbol{e}(t)||^2} \tag{39.5}$$

The mean squared error is further minimized by rescaling $\boldsymbol{x}_{est}(t)$ and adding an offset in each step and in the initial estimate.

Deconvolution results of the example are shown in Fig. 39.4. The original image was deconvolved with a 1-D Gaussian

$$h(z) = 1.65 \exp\left(-z^2/18\right) \tag{39.6}$$

The filter kernel contained 15 samples. The threshold was $\theta = 9$. In each iteration, only 3.44039 % of the image was filtered. The initial error was $E_{rms} = 1.276$. After three iterations it reduced to 0.868. Further reduction of the error is possible, but with irrelevant qualitative improvements. The reason for the restriction to a 1-D PSF was that resolution in z-direction can not be increased beyond that in x- and y-directions, and isotropy is important for the description of shape.

39.4 Differential feature detection for segmentation

As stated in Section 39.2, dye density is regarded as constant. Although this is a binary concept, intermediate intensities at locations inside an object are possible depending on object thickness in relation to the size of the PSF. Even after deconvolution, some blurring will remain due to noise suppression and isotropy. Segmentation by a *global threshold* will thus be incorrect. Instead, the threshold should be varied or a global threshold should be applied to a proper function of the gray values. We use the latter method.

The remaining blur is assumed to be *Gaussian*. As spine necks may be much thinner than the PSF thereof, segmentation must consider the geometry of the gray-level function. Watzel et al. [11] have devised a method of *segmentation* by extracting features from the *Taylor series expansion* of the gray-level function. This method works with objects that are thin with respect to the PSF. Ideas are discussed for 1-D signals first and are then extended to 3-D.

Consider a 1-D cut through a thin spine neck. The spine may be regarded as an impulse with an amplitude proportional to its cross-section area. The shape of the system response is nearly independent of the neck width and is approximated by a scaled Gaussian function. For a continuous 1-D thin object of width W, the system response is approximated by

$$\int_{-W/2}^{W/2} \exp\left(-\frac{(x-\tau)^2}{2\sigma^2}\right) \, d\tau \approx W \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{39.7}$$

Prior to thresholding the gray-level function must be described by a feature that is greater than some threshold θ inside objects and less outside. It is considered sufficient that the connectedness of the segmented objects is equivalent to that of the original objects and object centers lie inside the segmented regions. If $\theta = 0$, $-f''(x)$ will work satisfactorily, but for $\theta > 0$, an *amplitude-invariant feature* is desired. This can not be achieved in general because of superposition of various objects, but it is at least possible in ideal cases. The following features have been investigated:

$$h_1(x) = -\frac{f''(x)}{f(x)}, \quad h_2(x) = -\frac{f''(x)}{\kappa + |f'(x)|} \tag{39.8}$$

where κ is a small constant to prevent the denominator from being zero. Let us assume that there is a single object with response

$$f(x) = k \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{39.9}$$

For $\kappa \approx 0$, this yields

$$h_1(x) = -\frac{x^2 - \sigma^2}{\sigma^4} \quad \text{and} \quad h_2(x) \approx \frac{1}{|x|} - \frac{|x|}{\sigma^2} \tag{39.10}$$

While h_1 is independent of the amplitude k, h_2 is only for $\kappa \approx 0$. Its dependency of k decays with increasing distance from the center and can be controlled by the choice of κ. h_1 results in rather thick regions in comparison to the impulse assumed as an object. Also, both features can be corrupted by additive contributions to $f(x)$ from other objects.

The forementioned considerations have assumed a single object. Now consider two thin objects, one of which scaled by a factor k. Let them be located at positions δ and $-\delta$. Superposition of their system responses yields

$$f(x) = \exp\left(-\frac{(x+\delta)^2}{2\sigma^2}\right) + k\exp\left(-\frac{(x-\delta)^2}{2\sigma^2}\right) \tag{39.11}$$

The second-order derivative is

$$\begin{aligned} f''(x) &= \frac{1}{\sigma^2}\left(\frac{(x+\delta)^2}{\sigma^2} - 1\right)\exp\left(-\frac{(x+\delta)^2}{2\sigma^2}\right) \\ &+ \frac{k}{\sigma^2}\left(\frac{(x-\delta)^2}{\sigma^2} - 1\right)\exp\left(-\frac{(x-\delta)^2}{2\sigma^2}\right) \end{aligned} \tag{39.12}$$

Under certain conditions, there are two regions with $f''(x) < 0$, which surround the object centers. The first condition is $\delta > \sigma$, which guarantees that $f''(0) > 0$, and a threshold of $\theta = 0$ will separate the objects. Even if this is violated, some threshold $\theta < 0$ may still provide useful results in many cases.

The second condition requires that the contribution of one object should not spoil the other object at its center, thus $f''(\delta) < 0$. This implies

$$k > \left(4\frac{\delta^2}{\sigma^2} - 1\right)\exp\left(-2\frac{\delta^2}{\sigma^2}\right) \tag{39.13}$$

In the worst case ($\delta = 0.5\sqrt{3}\sigma$), this means $k > 2\exp(-3/2) \approx 0.446$. It can not be guaranteed that input data comply with these conditions. We conclude that the superposition of multiple object responses may still lead to *incorrect segmentation*. Even if the conditions are fulfilled, objects may be shifted and changed in size.

The concept of derivatives can be extended to three dimensions by Taylor series expansion of the gray-level function

$$f(x_1, x_2, x_3) = f(0,0,0) + \boldsymbol{g}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \ldots \tag{39.14}$$

Table 39.1: *Classification of local shapes*

λ_1	λ_2	λ_3	Class number	Description
$< \lambda_2$	$< \lambda_3$	< 0	1	near a point object
$< \lambda_2$	< 0	≥ 0	2	near a thin line object
				or far from a point object
< 0	≥ 0	$\geq \lambda_2$	3	near a thin plane object
				or far from a line object
≥ 0	$\geq \lambda_1$	$\geq \lambda_2$	4	outside any object

Table 39.2: *Class numbers of the directions of the eigenvectors $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$*

No.	Direction of $\boldsymbol{v}_2$	Direction of $\boldsymbol{v}_3$
1	tangential to the point	to the center of the point
2	to the center of the line (close)	parallel to the line (close)
	or tangential to the point (far)	or to the center of the point (far)
3	parallel to the plane	parallel to the plane
	or parallel to the line (far)	or to the center of the line (far)
4	irrelevant	irrelevant

where x_1, x_2 and x_3 are the image coordinates with respect to some point under examination; $\boldsymbol{g}$ is the *gradient* and $\boldsymbol{A}$ is the *Hessian matrix*. A *rotation-invariant description* can be obtained by transforming the Taylor series into the *eigensystem* of $\boldsymbol{A}$. In this system, the *eigenvalues* λ_i are the second-order derivatives in the direction of the associated *eigenvectors* $\boldsymbol{v}_i$. We order them such that $\lambda_1 \leq \lambda_2 \leq \lambda_3$. They can be used to classify the local shape similar to that in Chapter 13. Particularly, we are interested in the classes defined in Table 39.1.

To apply the 1-D concepts of Eq. (39.8), it is necessary to find the direction to the center of an object. If there is only one single object, the eigenvectors $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$ corresponding to λ_2 and λ_3, respectively, will indicate the directions in Table 39.2 depending on the shape of the object.

It is known that *dendrites* and *spines* are line-shaped objects, and spine heads can be modeled as point objects. This knowledge allows for suppression of plane objects by considering only classes 1 and 2. Therefore, we only evaluate Eq. (39.8) in the directions of $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$. In this way, many artifacts generated by deconvolution with a Gaussian PSF instead of the exact one are eliminated.

Sorting the eigenvalues such that $\lambda_1 \leq \lambda_2 \leq \lambda_3$ requires that derivatives be comparable. Because the standard deviation of the PSF is

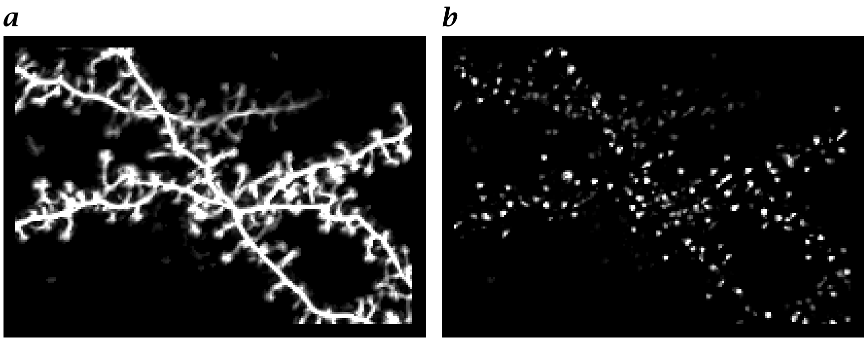


Figure 39.5: *Results of:* ***a*** *line filter (direction* $\boldsymbol{v}_2$*); and* ***b*** *point filter (direction* $\boldsymbol{v}_3$*) using* $h_2(x)$.

greater in z-direction than in the xy-plane, z-coordinates are scaled by an empirical factor of 0.7 when computing derivatives.

A problem arises at points close to a point object because $\boldsymbol{v}_2$ will be tangential, forcing Eq. (39.8) to behave as in the center of an object. This causes spine heads and other point objects to appear significantly larger than they are. On the other hand, in the direction of $\boldsymbol{v}_3$, spine heads appear close to the size of the PSF. This is still to large, but we regard it to be sufficient for *segmentation* because point objects are well separated. Therefore, results in both directions will be passed to the higher-level image analysis. Depending on the direction, the operation is called line filter or point filter, respectively.

Figure 39.5 shows the output of line and point filters using h_2 with $\kappa = 10$. For further segmentation, the threshold is chosen $\theta = 25 \pm 5$, and it is adjusted manually to reduce the remaining errors in connectedness.

The width of thin objects in the output is nearly independent of their true width as can be seen in Eq. (39.8). Instead, it depends on the size of the PSF. Therefore, measurements of thickness will require information from the original gray-level image.

39.5 Topology preserving three-dimensional thinning

The purpose of the forementioned preprocessing steps is to give a measure of whether an image point should be segmented as object or background. The results obtained are segmented using a global threshold. Ideally, the binary result should be *topology* equivalent to the real tissue and it should preserve its relevant features. The former can not of course be guaranteed because reconstruction can not be perfect. Topological error may be corrected in higher stages of the process when shape understanding is possible. As a link to higher stages, the objects

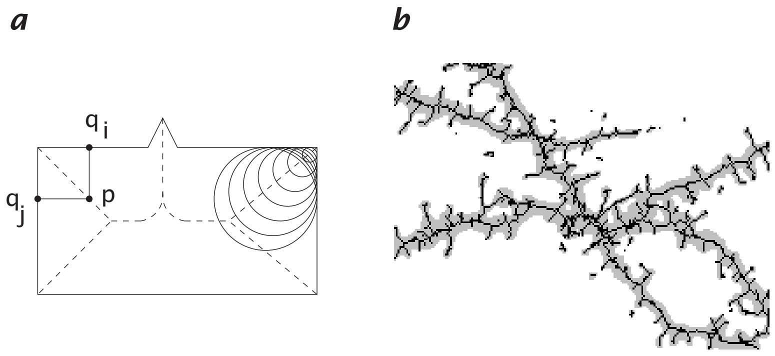


Figure 39.6: ***a*** *Illustration of the concept of a skeleton; and* ***b*** *a skeleton of the example image (deleted regions are marked gray).*

are reduced to line skeletons. Because there is no scene understanding at this stage, the skeleton is to preserve the topology as segmented. The algorithm used here is described by Watzel et al. [17]. Related works can be found in [18, 19], and [20].

A *skeleton*, which is also known as a *medial axis transform*, comprises the subset S of all points of an object O with boundary ∂O such that $S = \{p | p \in O \wedge \exists q_i, q_j \in \partial O : q_i \neq q_j \wedge d(p, q_i) = d(p, q_j) \wedge \forall q_k \in \partial O : d(p, q_j) \leq d(p, q_k)\}$, where d is a distance measure as described in Volume 2, Section 2.3.4. Figure 39.6a gives a 2-D example.

As indicated in Fig. 39.6a, the shape can be reconstructed from the skeleton S exactly if at every point in S a circle is drawn with maximum radius. The mapping from S to such radii is called the *radius function*.

Figure 39.6a also illustrates that a small edge at the object boundary may cause a skeleton branch that only represents a minor feature of shape. Such branches are called spurious, and their number can be reduced by smoothing object boundaries using such operations as opening and closing (Volume 2, Chapter 21).

In digital images, the concept of skeleton is difficult to define because in general the center lines are not a subset of the grid. Nevertheless, there exist a number of algorithms to compute digital skeletons. However, an overview would be beyond the scope of this section. Here, we only give a rough sketch of the applied thinning algorithm.

In the following, the set of *voxels* inside objects is referred to as foreground, and such voxels are referred to as black points. The remaining voxels are called background and are referred to as white points. For the foreground, 26-*adjacency* is used, and for the background 6-adjacency is used (Volume 2, Section 2.3.5). The set of all points within the 26-*neighborhood* of some point p is denoted as $N(p)$. The *thinning algorithm* works as follows:

1. Define an ordered set of directions $R = \{r_0, \ldots r_5\}$ that lie in the cubic grid. Select r_0. Set $i = 0$.
2. A point whose 6–neighbor in direction r is white is called *r-free*.
3. Let B denote the set of all black points. Determine the set $M \subseteq B$ of all r–free black points.
4. Determine the set of points $D \subseteq M$ of deletable points (for deletability, see text following Step 6).
5. If $D = 0$, terminate.
6. Replace B by $B \setminus D$, set $i = (i + 1) \quad mod \quad 6$ and select direction r_i. Continue with Step 3.

The key point in the foregoing algorithm is the definition of deletability. First, corner points must be preserved because they comprise relevant features of shape. If a point is deleted, it can not be reconstructed via the radius function. Second, the connectedness of objects and background must be preserved. For details, the reader is referred to the literature.

In 3-D, by definition, the skeleton may include medial faces. But even these are reduced to lines by the forementioned algorithm because of the choice of our deletability criterion. This agrees with the assumption that all expected objects have circular cross sections. The series of directions in the thinning algorithm implies that centers are found with respect to city-block distance; see Volume 2, Chapter 2.3.4. However, for the computation of the radius function, Euclidean distance is used to improve isotropy.

39.6 Graph construction and interpretation

It is difficult for a computer to perform shape analysis on a skeleton as described in Section 39.5 because the skeleton consists only of an improperly ordered set of points. To represent global structure, the skeleton is translated into a graph $\mathcal{G}_1$ by a line-traversing algorithm. This algorithm first generates an ordered set of voxels $V = (v_1, v_2, \ldots)$. Each voxel is encoded as a tuple $v = (x, y, z, r, l)$, where x, y and z denote the coordinates of the voxel, r is the value of the *radius function* and l is a label. Voxels that constitute an unbranched line segment are ordered such that each one is connected to its neighbors. Members of such a line have identical label l. Different line segments have different labels.

A graph comprises a tuple $\mathcal{G} = (\mathcal{E}, \mathcal{V})$, where $\mathcal{E}$ denotes the set of edges and $\mathcal{V}$ the set vertices. End points and junction points of the skeleton will be represented by vertices, while the connecting lines are encoded as *edges*. A *vertex* consists of a tuple $\boldsymbol{v} = (x, y, z, \mathcal{E}_{\boldsymbol{v}}, l_1, l_2)$, where x, y and z are the coordinates of the corresponding image point;

l_1 and l_2 are labels to store semantic properties; $\mathcal{E}_v$ denotes a set of references to edges that connect $\boldsymbol{v}$ to other vertices. Vertices are generated for each voxel in V at which label l changes.

Edges are encoded as tuples $\boldsymbol{e} = (\boldsymbol{v}_1, \boldsymbol{v}_2, i_1, i_2)$, where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ reference the vertices to be connected, and i_1 and i_2 reference the beginning and end of a line segment in V. Edges are generated for each line segment. (For a more detailed review of graph theoretical concepts, see Volume 2, Chapter 24.)

According to the description of dendrites and spines in Section 39.1, it is a natural approach to identify these objects by their length. Length information is consistently incorporated in the graph by the concept of height, which is originally defined for a tree [21]. In a *tree*, the *height* of a vertex $\boldsymbol{v}$ is the longest distance from $\boldsymbol{v}$ to a *leaf*. This definition holds because a tree is a directed *acyclic graph* with known root. Here we measure distance by summing the length of all line segments on the path from $\boldsymbol{v}$ to the leaf. We know neither edge directions nor the root, but we can construct them by computing the height dynamically from bottom up (from the leafs to inner vertices). The *root* results as the vertex of maximum height.

Height computation requires that the graph be acyclic. We guarantee this by opening each cycle at the point where the radius function has a minimum. This is a heuristic incorporating the idea that the thinnest part is most unlikely to belong to the dendrite. After height computation, cycles are restored for further analysis.

The dendritic structure is identified in the graph by traversing the graph from the root by following paths of descending height. Branches will be traversed if their height exceeds a properly chosen threshold ($3\,\mu$m < threshold < $8\,\mu$m).

Now the shape of the dendritic structure is reconstructed from the radius function by changing labels in the label image from which the graph has been constructed. Points in the inside of the dendrite are labeled with a specific value. Skeleton points are labeled with a different value. Also, point objects obtained from the methods described in Section 39.4 are labeled in this image, with different values for skeleton points. This is not done inside the dendritic structure. From this label image, a new skeleton is computed preserving information about dendrite and head regions by setting proper labels.

From the result a new graph $\mathcal{G}_2$ is constructed. At this time, vertices are also generated at points at the border between different labels. In $\mathcal{G}_2$ the identification of dendrites is restricted to regions associated with the dendrites found in $\mathcal{G}_1$. The generation of the second graph allows for different segmentation techniques for dendrites, spine necks and heads. Finally, $\mathcal{G}_2$ is used to construct a list of spine candidates.

Spine candidates are identified by determination of base points and traversal of the nondendritic structures attached to them. All vertices

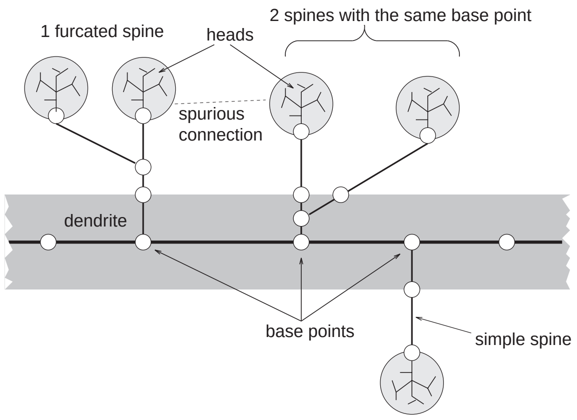


Figure 39.7: *Interpretation of a graph. Spines are regarded as the nondendritic structures attached to base points. According to previous segmentation, subgraphs are marked as dendritic, nondendritic or head.*

in a dendritic path that have nondendritic edges are considered as base points (Fig. 39.7). The spine-traversal algorithm uses the following rules:

1. From a base point, multiple spines may originate.
2. If the spine structure furcates inside the dendrite, the branches are regarded as distinct spines.
3. If the spine structure furcates outside the dendrite, the structure is regarded as one furcated spine.
4. If the algorithm enters a head, it will never leave it, avoiding spurious connections between heads.

Rule 4 is important because it combines information from the line and the point filter. Spine heads are increased dramatically in size by the line filter. The rule restricts the traversal to the size determined by the point filter.

39.7 Results

From the graph depicted in Fig. 39.8, six dendritic branches have been extracted. Their graphs are depicted in Fig. 39.9 as independent objects. For each of the branches, a list of spine candidates has been generated. For each spine its length has been measured in its graph. Table 39.3 shows the list of dendrites with their measured lengths and the number of spines. The spine lists have been cleaned using a length threshold of 0.2 μm to delete spines that originate from spurious skele-

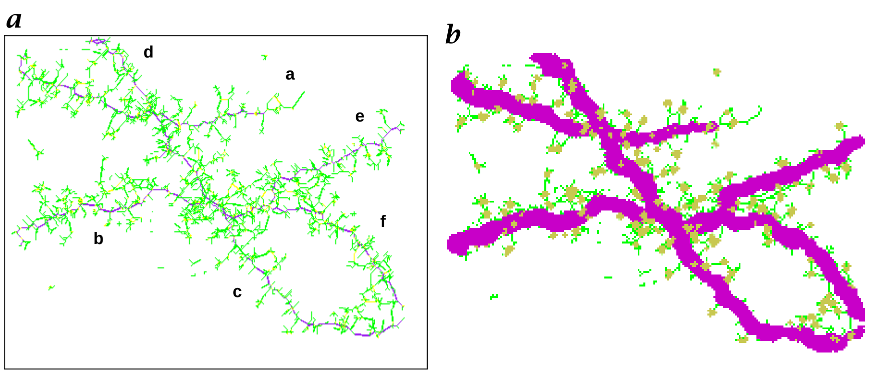


Figure 39.8: ***a*** *Graph obtained from the example. The letters refer to the dendritic branches as shown in Fig. 39.9;* ***b*** *skeleton after reconstruction of dendrites and spine heads.*

Table 39.3: *List of dendrites with spine numbers*

Dendrite number	Length / μm	Raw number of spine candidates	Number of spines cleaned automatically	Number of spines cleaned manually
a	14.27	146	70	66
b	12.63	71	31	30
c	21.94	168	86	86
d	24.20	150	82	77
e	13.21	89	44	39
f	21.54	87	56	51

ton branches and a threshold for dendrite radius of 0.2 μm. The resulting list has been further cleaned manually by consideration of spine length and spine position with regard to the dendrite end points. Figure 39.10 shows the length histogram of the manually cleaned spine list.

39.8 Discussion

Many of the problems addressed here arise from the limited *optical resolution* of the image acquisition equipment and from the small size of the targets of interest. If the shape of the targets could be observed correctly, it would be sufficient to compute a *skeleton* and a *graph*, and spines could be identified in the graph reliably. Unfortunately, correct restoration of true shape is impossible because of *blurring*, the

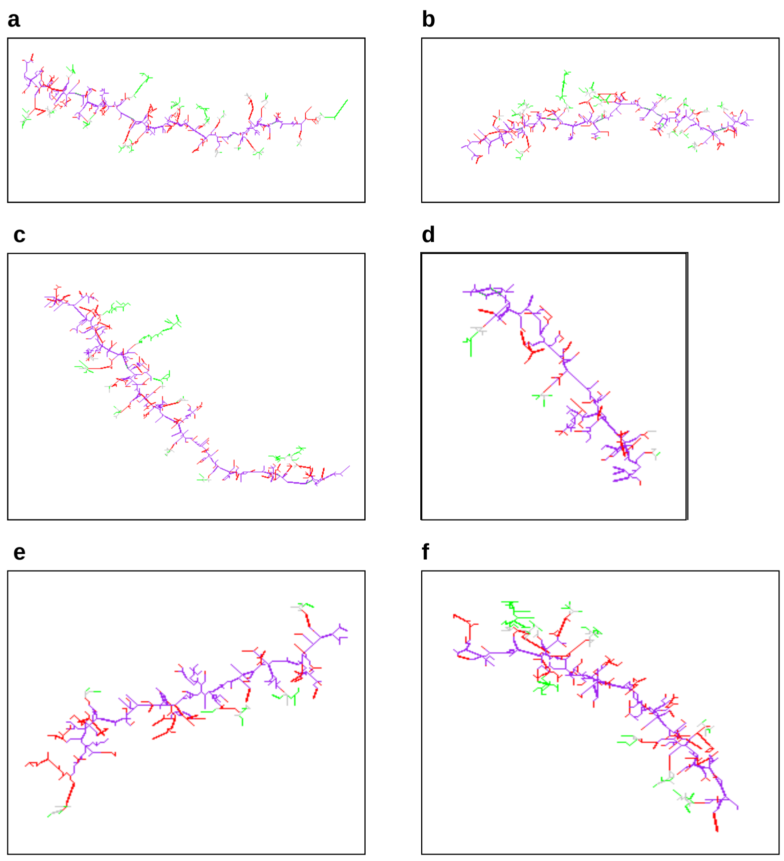


Figure 39.9: Dendritic subgraphs extracted from the example.

presence of noise, and *undersampling*. In addition, it has been observed that the PSF is shift-variant and deviates from its theoretical shape. The deconvolution method used here was selected to achieve adequate results in short computation time.

The differential methods are used because of the special condition that targets are thin in comparison to the PSF. They allow for distinction of line- and point-shaped objects and for a nearly intensity-invariant description of their system response. They come with the drawback that object width is replaced by PSF size, causing problems in measurement of radii. We think that these methods may also be used to find skeletons. At least, they can thin an object before *segmentation*, and a succeeding binary *thinning* may find the final skeleton. This may further reduce the number of spurious branches and improve the medialness of the skeleton.

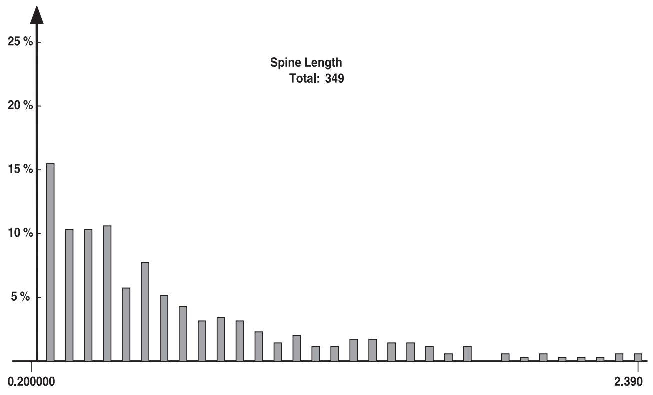


Figure 39.10: *Length histogram of all spine candidates.*

The notion of point-shaped objects has been incorporated into the graph analysis. This reduces errors for the frequent case of spine heads merging due to close distance. However, spine necks in close proximity and furcated spines as well may cause point-shaped regions at locations where no head exists. In these cases spines are shorter than they really appear. More sophisticated graph analysis will have to consider a large variety of decision alternatives regarding which branch belongs to which spine, and it will have to find the best solution according to some likelihood function. As this comprises considerable effort and no improvement can be guaranteed, it is currently beyond the scope of this work.

For the description of shape, it would be desirable to measure the surface or volume of the spines found. However, we expect such results not to be reliable because of the great uncertainty induced by image degradation. In this chapter we considered only measurement of length because we regard this as the most reliable measurement.

Acknowledgment

This project was supported by the Deutsche Forschungsgemeinschaft from 1994 to 1997 at the Leibniz Institute for Neurobiology, registered as INK 15/A1. Later, one of the authors, Watzel, received support by the Department of Computer Engineering at Darmstadt University of Technology to continue this research and to finish his PhD thesis.

39.9 References

[1] Bock, J., (1998). *Lerninduzierte synaptische Veränderungen in assoziativen Vorderhirnarealen des Haushuhnkükens (Gallus gallus domesticus): metabolische, morphologische und pharmakologische Untersuchungen unter besonderer Berücksichtigung des NMDA-Rezeptors.* PhD thesis, Otto-v.-Guericke University Magdeburg.

[2] Braun, K., Bock, J., Metzger, M., and Schnabel, R., (1998). The dorso-caudal neostriatum of the domestic chick: a structure serving higher associative functions. *Behav. Brain Res. in press.*

[3] Scheich, H., Wallhäuser, E., and Braun, K., (1991). Does synaptic selection explain auditory imprinting? In *Memory: Organization and Locus of Change*, L. Squire, J. Weinberger, E. Lynch, and N. McGaugh, eds., pp. 114–159. Oxford: Oxford University Press.

[4] Braun, K., (1998). Plastizität der Gehirnbiologie bei frühkindlichen Lern- und Erfahrungsprozessen: Relevanz für die Entstehung psychischer Erkrankungen. In *Im Spannungsfeld zwischen Generalisierung und Integration—Perspektiven der Psychiatrie, Psychotherapie und Nervenheilkunde*, W. Gaebel and P. Falkai, eds., pp. 4–9. Wien, New York: Springer.

[5] Faber, H., (1992). *Ultrastrukturelle Konsequenzen des akustischen Prägungslernens auf identifizierte Neurontypen im Telencephalon von Haushuhnküken (Gallus gallus domesticus).* Dissertation, Technische Universität Darmstadt.

[6] Watzel, R., Braun, K., Hess., A., Scheich, H., and Zuschratter, W., (1995). Detection of dendritic spines in 3-Dimensional images. In *17. Symposium der Deutschen Arbeitsgemeinschaft Mustererkennung (DAGM)*, pp. 160–167. Berlin: Springer.

[7] Watzel, R., Braun, K., Hess, A., Zuschratter, W., and Scheich, H., (1997). Computer aided analysis of dendritic spines. *25th Goettingen Neurobiology Conference*, p. 1050.

[8] Frieden, B. R., (1975). Image enhancement and restoration. In *Picture Processing and Digital Filtering*, T. S. Huang, ed., chapter 5, pp. 177–248. Berlin: Springer.

[9] Conchello, J.-A. and Hansen, E. W., (1990). Enhanced 3-D reconstruction from confocal scanning microscope images. 1: Deterministic and maximum Likelihood reconstructions. *Applied Optics*, **29**(26):3795–3804.

[10] White, R. L., (1994). Image restoration using the damped Richardson-Lucy method. In *The Restoration of HST Images and Spectra II*, R. J. Hanisch and R. L. White, eds., pp. 104–110. Space Telescope Science Institute.

[11] Watzel, R., Braun, K., Hess, A., Zuschratter, W., and Scheich, H., (1996). Restoration of Dendrites and Spines with the Objective of Topologically Correct Segmentation. In *13th International Conference on Pattern Recognition (ICPR), Vienna*, pp. 472–476.

[12] Thirion, J.-P., (1993). *New feature points based on geometric invariants for 3D image registration.* Technical Report, INRIA Research Report RR-1901.

[13] Thirion, J.-P. and Benayoun, S., (1993). *Image surface extremal points, new feature points for image registration.* Technical Report, INRIA Research Report RR-2003.

[14] Sander, P. T. and Zucker, S. W., (1990). Inferring surface trace and cifferential structure from 3-D images. *IEEE Trans. PAMI*, **12**(9):833–854.

[15] Sander, P. T. and Zucker, S. W., (1992). Singularities of principal direction fields from 3-D images. *IEEE Trans. PAMI*, **14**(3):309–317.

[16] Bigün, J. and Granlund, G. H., (1987). Optimal Orientation Detection of Linear Symmetry. In *First International Conference on Computer Vision (London, England, June 8–11, 1987)*, pp. 433–438. Washington, DC.: IEEE Computer Society Press.

[17] Watzel, R., Braun, K., Hess, A., Scheich, H., and Zuschratter, W., (1995). On the deletability of points in 3D thinning. In *International Computer Science Conference*, pp. 91–98. Berlin: Springer.

[18] Ma, C. M., (1996). Connectivity Preservation of 3D 6-Subiteration Thinning Algorithms. *Graphical models and image processing: GMIP*, **58**(4):382–386.

[19] Tsao, Y. and Fu, K., (1981). A parallel thinning algorithm for 3D pictures. *Computer Graphics and Image Processing*, **17**:315–331.

[20] Lee, T.-C. and Kashyap, R. L., (1994). Building skeleton models via 3-D medial surface/axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, **56**(6):462–478.

[21] Aho, Hopcroft, and Ullman, (1974). *The Design and Analysis of Computer Algorithms.* Reading, MA: Addison-Wesley.

40 Principles of Spectral Precision Distance Confocal Microscopy for the Analysis of Molecular Nuclear Structure

Christoph Cremer[1,2], P. Edelmann[1,2], H. Bornfleth[1,2], G. Kreth[1,2,3], H. Muench[1], H. Luz[1], M. Hausmann[1]

[1]Institut für Angewandte Physik
[2]Interdisziplinäres Zentrum für Wissenschaftliches Rechnen
[3]Graduiertenkolleg, Wissenschaftliches Rechnen
Universität Heidelberg, Germany

40.1 The problem

Localization by light plays a fundamental role in many fields of science. In optics, during the last century, the development of high-performance light microscopes was at the foundation of a revolution in biology and medicine. In combination with appropriate staining techniques, light microscopy allowed the discovery of the elementary unit of all organisms, the cell. Further light microscopical analysis of the cellular structure soon resulted in the discovery of a main constituent of the cell, the cell nucleus, and of the chromosomes ("stained bodies"). At that time, the latter were visible during cell division only but they had to be present in some unknown form also in the nucleus. Today we know that the chromosomes contain (with very few exceptions) all the information necessary for cellular metabolism and for the growth and development of every organism. Together, they form the genome that in normal human cells is divided into 46 individual chromosomes.

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

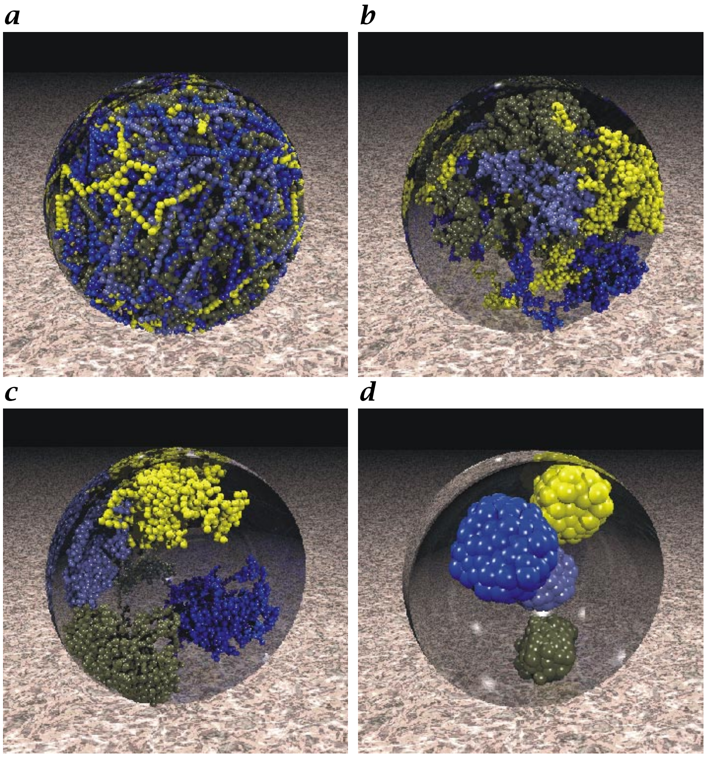


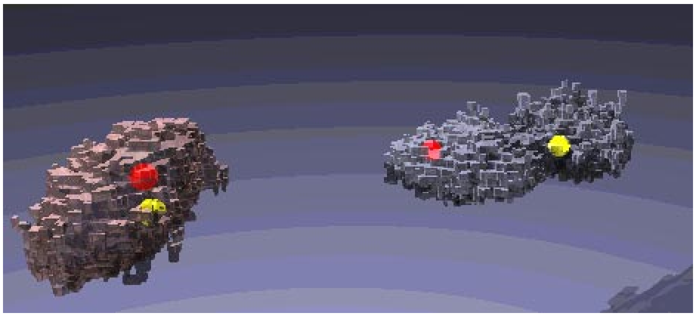
Figure 40.1: *Computer simulations of all chromosomes in the human interphase cell nucleus. Chromosomes were described by different models, which approximate the chromatin fiber by a polymer chain, folded in different ways. Corresponding to the different condensation levels, a nonterritorial (**a** Vogel and Schroeder-, **b** random-walk bead model) or a territorial (**c** random-walk giant loop-, **d** spherical subdomain model) organization of chromosomes was obtained. For visualization, in each case only four chromosomes are visualized in a spherical nuclear envelope by ray tracing. For further details see Kreth et al. [1]; (see also Plate 15).*

The chromosomes are the most complex molecules found so far in nature. For example, each human chromosome is formed from a single DNA-molecule with a specific sequence of base pairs carrying the hereditary information and consisting of several 10^9 atoms; the 50 to 230 million base pairs of a human chromosome are further associated in a specific way with about several million proteins, the histones. This enormous complexity, however, is not visible by conventional light microscopical procedures. Thus, for more than a century discussions of even very fundamental questions concerning the structure of the chromosomes, especially in the cell nucleus, were controversial. For example, light and even electron microscopical observations as well as biochemical data were compatible with models where DNA-protein-fibers of each of the chromosomes were stretched out through the entire nucleus but also with models where these fibers were folded in such a way that the chromosome had an enveloping volume of a few percent of the entire nuclear volume only (for a review, see [2]). As an example, Fig. 40.1 shows visualizations of such models obtained by computer simulations of all chromosomes in the human interphase cell nucleus [1, 3, 4, 5, 6].

More than half a century ago, Erwin Schroedinger formulated a hypothesis that was to become decisive for the development of molecular biology. He speculated that the chromosomes in the nucleus were “aperiodic solid bodies” with a complex 3-D structure, and that this was the key to their high informational content. Today, we know that this hypothesis was completely correct on the level of individual base pairs and of small clusters of so-called nucleosomes of 11-nm diameter (1 nm = 1×10^{-9} m), each with about 200 base pairs (including “linker” DNA to neighboring nucleosomes) associated with eight histone proteins in a very precise manner. Recently, it has become possible to unravel the 3-D structure of the nucleosomes with almost atomic resolution by synchrotron radiation diffraction experiments. From a formal point of view, this reduced the problem of the molecular nuclear structure to the problem of finding the relative 3-D positions for each nucleosomal DNA sequence. For a human cell nucleus, this would mean the topological analysis of about 35 to 70 million different sites. Until recently, even partial solutions of this enormous problem (e.g., relative positions of selected nucleosome clusters, or the morphology of still larger units) appeared to be impossible.

Since the 1970s, the progress of molecular biology opened an avenue to the labeling of a variety of targets in the human cell nucleus in a specific way using fluorochrome molecules. For example, in principle, any known protein type or any known DNA sequence of the genome can now be individually fluorescence labeled. In a simple, schematic form this technique called “fluorescence in situ hybridization” (FISH) is based on the binding of a fluorescence-labeled “probe” DNA to the complementary sequence of the single-stranded (“denatured”) chromosomal “target” DNA “in situ,” that is, directly in the cell nuclei [7]. Using fluorochromes of different spectral signature (characterized, e.g., by different absorption or fluorescence emission spectra), the chromosomal target DNA sequences can be labelled in a spectrally distinct way.

As an example, Fig. 40.2 shows a 3-D reconstruction of a human female cell nucleus where the two female sex chromosomes (the X) plus 2 genes, located on them, were FISH-labeled with different spectral signatures and registered by confocal laser scanning fluorescence microscopy (CLSM). For a quantitative image analysis of shape parameters, both Voronoi and Cavalieri estimators have been successfully



X_r	X_e
SF = 0.910	SF = 0.354
RF = 0.088	SF = 0.008
Volume = 18.94 μm^3	Volume = 16.39 μm^3
Surface area = 77.07 μm^3	Surface area = 154.84 μm^3

Figure 40.2: *Reconstructed confocal 3-D dataset of a human female cell nucleus visualized by ray tracing. The genes ANT2 (red)/ ANT3 (yellow) are represented by spheres with approximately the same volume as the registered signals in the confocal image. The rounder shape of the inactive X-chromosome territory (Xi, red) and the more interior localization of the inactive ANT2 on Xi is clearly visible (SF = smoothness factor; SF converge against 0, elongated shape; SF converge against 1, spherical shape; RF = roundness factor; see Bornfleth et al., Chapter 41.2).*

applied. It was even found that the same chromosome territory may have a distinctly different morphology according to its genetic activity [8, 9]. Compared with the extension of the entire nucleus, the individual chromosome territories clearly have an enveloping volume of a few percent of the entire nuclear volume only. The "territoriality" of chromosomes has now been firmly established as a main feature of nuclear organization of normal cells; it excludes a variety of models as general descriptions of chromosomal structure (compare Fig. 40.1).

Taken by itself, the territoriality of chromosomes in the cell nucleus (in the sense of a limited extension) may be quantitatively modeled by relatively simple polymer or "bead" models (compare Fig. 40.1; [1, 4, 6, 10]). Even light microscopically measured distances between individual, FISH-labeled DNA target sequences within the same chromosome territory were estimated by such models in a quantitatively satisfying manner. An alternative view holds, however, that due to a "chromatin folding code," a functionally important degree of higher-order structures of the chromosome is obtained (for review, see [2, 4, 11]). To test such a hypothesis of considerable impact for our understanding of life, a much more detailed quantitative analysis of nuclear structure is necessary; this appears to be especially meaningful for the molecular genome structure below about 100,000 base pairs (100 kbp) of DNA (about 500 nucleosomes including “linker” DNA). From the microscopical point of view, this requires measuring 3-D geometric distances and spatial conformations of FISH-labeled target sequences located on the same chromosomal fiber that have a linear genomic distance of less than about 100 kbp. Experimental evidence, however, indicates that using advanced far-field light microscopical methods such as confocal laser scanning microscopy (CLSM), geometrical target 3-D distances in the nucleus below several hundred nanometers (corresponding to genomic distances below about 100 kbp) cannot be measured reliably if both target sequences are labeled with the same spectral signature. This is due to the well-known limits of microscopical resolution (Volume 1, Chapter 21; [12]).

Figure 40.3 shows an computed example for the limited resolution of CLSM for conditions relevant in genome structure analysis. The simulations made on the basis of scalar wave theory using point objects of one spectral signature indicate that even under the (unrealistic) assumption of an ideal photon statistics and lack of detector noise, a 3-D resolution of better than a few hundred nanometers is hardly feasible. Under “practical” conditions of biological routine measurements, the 3-D resolution of CLSM (as determined from the axial full width at half maximum (FWHM) of the confocal point-spread function (PSF)) was found to be on the order of 600 to 700 nm [13]. As an example of the consequences of the limited resolution in genome structure research, Fig. 40.4a shows the simulated projection image of a 70-kbp long DNA sequence of a chromosomal fiber (calculated as a zig-zag random walk of nucleosomes) after labeling with one spectral signature and convolution with a PSF of 250 nm FWHM. As expected, even a rotation of the 70-kbp sequence by axial tomography [14] under these conditions would not reveal any significant structural detail.

To increase the 3-D light optical resolution in the sense of point-spread function improvement, other microscopical far-field methods have to be considered, such as 4Pi microscopy [15, 16], standing-wave-field microscopy [17]), spatially modulated excitation (SME) microscopy [18, 19], and other techniques of point-spread function engineering. In the following, the principles of an additional approach to reveal 3-D genome structure on a scale considerably lower than the classical resolution limit will be considered. Generally, this approach called spectral precision distance microscopy (SPM) makes possible the measurement of distances well below the FWHM of the effective PSF that describes the optical properties of the imaging system; SPM combines high-quality optics with advanced techniques of computer image analysis and reconstruction. Because SPM can be used with any optical system having a well-defined PSF, it will greatly increase the possibilities of the

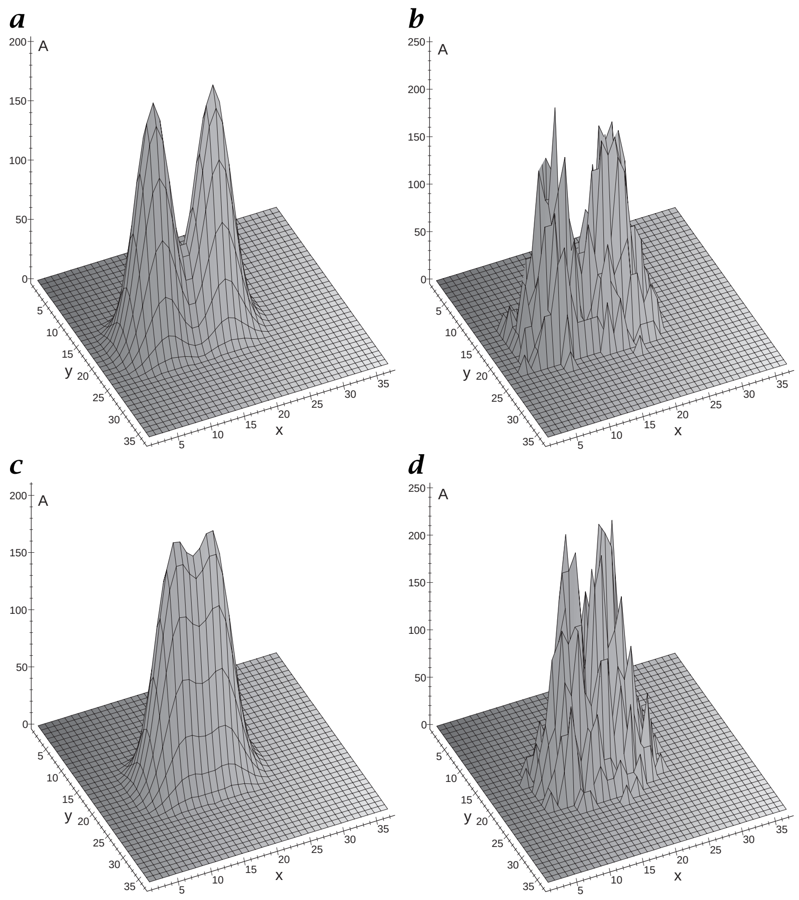


Figure 40.3: Computer-generated example for the resolution of the CLSM using two point-like targets of the same spectral signature of λ = 520 nm. The PSF was sampled with stepwidths of 0.4 optical units (27 nm): ***a*** *two point-like targets with a lateral distance of 226 nm, corresponding to the Rayleigh criterion (~ λ/NA) of optical resolution;* ***b*** *the same image after application of Poisson noise. A maximum number of 12 detected photons per voxel was assumed;* ***c***, ***d*** *two point-like targets with a spatial distance of 138 nm, corresponding to the FWHM criterion of optical resolution, before and after application of Poisson noise as in* ***b****.*

point-spread function engineered devices mentioned in the foregoing. In combination with digital microscopy and digital image analysis, SPM opens the possibility of measuring light microscopically 3-D positions of fluorescence labeled targets on a scale several to many times smaller than the nominal resolution of the system used. In combination with computer modeling of the regions studied and—where possible—with ultrastructural procedures, this will eventually allow the unraveling of the 3-D structure and texture of the genome down to the molecular level in an analogous way as it has become possible for other biological macromolecules such as large proteins.

40.2 Principles of spectral precision distance microscopy (SPM)

It has been known for many years that "super resolution" of point objects situated within the Rayleigh distance (corresponding approximately to 1 FWHM of the PSF) can be achieved if the two objects possess different spectral characteristics. Strategies of deconvolution where each image associated with a particular wavelength is separately deconvoluted with the transfer function of that wavelength have been described [20]. Even when the data were characterized by a moderate signal to noise ratio (SNR), simulated images of point-like objects were correctly restored that were only 1/30–1/50 FWHM apart from each other. The SPM approach described here [21, 22] is based on the same principle idea that using spectral information, the old resolution limits such as those of Rayleigh or Abbé are no longer valid. That principal idea has been extended, however, to the special needs of high-resolution fluorescence far-field microscopy and its application to the 3-D analysis of the molecular nanostructure of the genome in the mammalian cell nucleus.

The principle of SPM is shown schematically in Fig. 40.5: Let us assume three closely neighboring targets in a nucleus to be studied with distances much smaller than 1 FWHM, where FWHM represents the full width at half maximum of the effective PSF of the optical system used. The "point-like" (diameter much smaller than 1 FWHM) targets t_1, t_2 and t_3 (e.g., three genes, or two genes and one protein) are assumed to have been labeled with three different fluorescent spectral signatures $specs_1$, $specs_2$ and $specs_3$. For example, t_1 was labeled with $specs_1$, t_2 with $specs_2$ and t_3 with $specs_3$. The registration of the images (using, e.g., CLSM) is performed in a spectrally discriminated way so that in a first 3-D image stack IM_1, a $specs_1$ intensity value I_1 is assigned to each voxel v_k of the object space; in a second 3-D image stack IM_2, a $specs_2$ intensity value I_2 is assigned to each voxel v_k of the object space; and in a third 3-D image stack IM_3, a $specs_3$ intensity value I_3 is assigned to each voxel v_k of the object space. So far, the strategy is very similar to that described by Burns et al. [20] for one spatial dimension. Instead of wavelength-discriminating deconvolution algorithms, however, the SPM approach is based on the direct evaluation of the spectrally separated images. In ideal epifluorescent or 2Pi, 4Pi confocal images, for basic optical reasons the absolute maximum max_1 of the diffraction

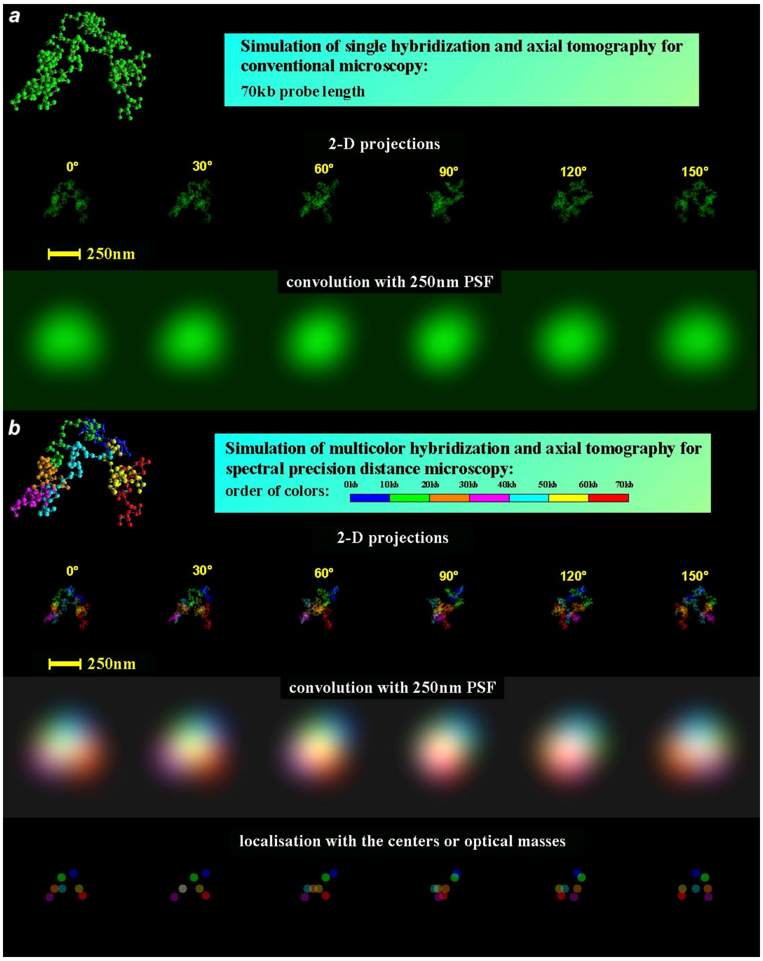

a
Simulation of single hybridization and axial tomography for conventional microscopy:
70kb probe length
2-D projections
0°
30°
60°
90°
120°
150°
250nm
convolution with 250nm PSF
b
Simulation of multicolor hybridization and axial tomography for spectral precision distance microscopy:
order of colors:
0kb
10kb
20kb
30kb
40kb
50kb
60kb
70kb
2-D projections
0°
30°
60°
90°
120°
150°
250nm
convolution with 250nm PSF
localisation with the centers or optical masses

Figure 40.4: ***a*** *Zig-zag random-walk computer simulation of a 70-kb long nucleosome chain (beads) of chromosomal DNA after single color hybridization. In conventional fluorescence microscopy (corresponding to an assumed convolution with a PSF of 250-nm FWHM), even under various rotation angles no inner structures are detectable;* ***b*** *zig-zag random-walk computer simulation of the same 70-kb long nucleosome chain of chromosomal DNA after hybridization with seven spectrally different signatures (each color segment 10 kbp). Each segment is assumed to be detected by spectrally discriminated imaging. In spectral precision distance microscopy (SPM), the barycenters of the intensities can still be localized after convolution with the PSF of the microscope, so that their distances can be measured. In combination with axial tomography for rotation of the labeled object, the 3-D conformation of the DNA segment can be revealed, although the barycenter distances were considerably below the resolution limit of the microscopy given by the FWHM of the PSF; (see also Plate 16).*

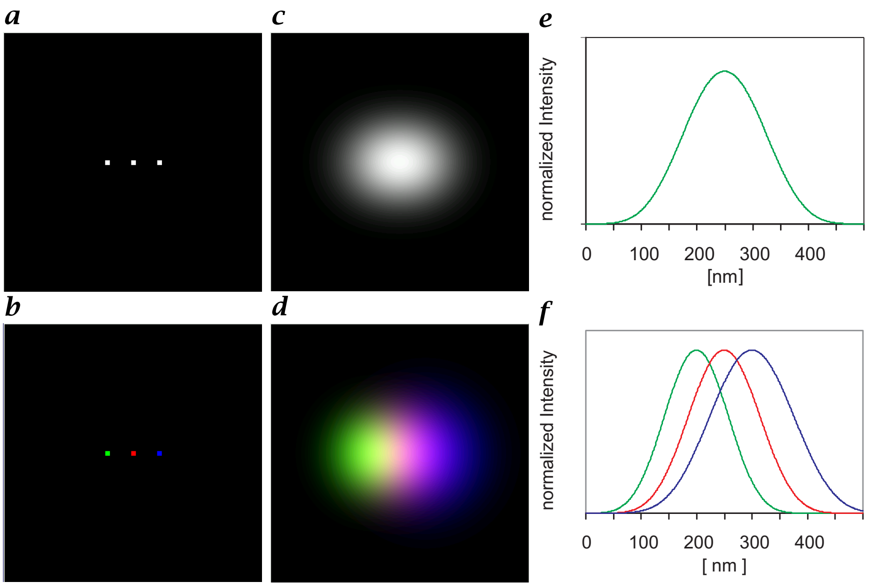


Figure 40.5: *Example describing the principle of spectral precision distance microscopy (SPM). Three point-like objects are located within 50-nm distance of each other. The three point-like objects are labeled with the same spectral signature in **a** and with three different spectral signatures in **b**. The computed system responses of the objects in **a** and **b** for a confocal microscope with NA = 1.4/63× oil-immersion objective are shown in **c** and **d**. Linescanes through the objects in **c** and **d** are shown in **e** and **f** respectively (for details see text and [23]); (see also Plate 17).*

pattern of the image of t_1 (registered as a specs$_1$ intensity distribution) gives the ideal (geometrical) image point i_1 of t_1; the absolute maximum max$_2$ of the diffraction pattern of the image of t_2 (registered as specs$_2$ intensity distribution) gives the ideal (geometrical) image point i_2 of t_2; and the absolute maximum max$_3$ of the diffraction pattern of the image t_3 (registered as specs$_3$ intensity distribution) gives the ideal (geometrical) image point i_3 of t_3. From this, the positions (object points) o_1, o_2 and o_3 of t_1, t_2 and t_3 and their 3-D distances in the object space can be determined if the optical characteristics of the system (such as magnification factors, aberration distortions) are known.

In principle, the minimum (real) distance that can be measured in this way is independent of the FWHM of the PSF and varies only with uncorrected systematic errors and statistical errors affecting the localization of the diffraction maxima. This minimum detectable distance between two point-like objects of different spectral signatures one may call the resolution equivalent (RE). It has to be distinguished from the optical resolution (A) that usually refers to the capacity of an optical system to transfer structural information (minimum distance detectable) of *any* object structure. In general, A is given, for example, by the Abbe-, Rayleigh-, Sparrow (or similar) criterium, the FWHM of the PSF, or the limiting spatial frequency of the optical transfer function (OTF). It is clear that the SPM strategy can be applied to more than two or three closely neighbored targets, if the neighboring targets t_1, t_2, t_3, ..., t_n have sufficiently different spectral signatures $specs_1$, $specs_2$, $specs_3$, ..., $specs_n$. Three or more spectral signatures allow true structural conclusions. For example, already three targets with three different spectral signatures allow us to define distance and angle relationships between the targets; four targets with four different spectral signatures allow a variety of 3-D structures.

Furthermore, it is clear that essentially the same SPM strategy can be applied also in all cases where the distance between targets of the *same* spectral signature is larger than FWHM [24]. Computer simulations performed by Bornfleth et al. [21] indicated that a distance of about 1.5 FWHM is sufficient. For example, in a nucleus with a volume of about $500\,\mu\text{m}^3$ and a confocal observation volume approximated by $V_{\text{obs}} = 4\pi/3\ (FWHMxy)^2 \times (\text{FWHMz}) = 0.13\,\mu\text{m}^3$, this means that about 10^3 different multispectral target sites can be evaluated in the same nucleus.

To realize local super-resolution strategies (RE much smaller than 1 FWHM), using spectral characteristics as constraints in a microscopically useful way, a variety of pitfalls have to be overcome. Here, based on our present experience, some fundamental problems of the realization of confocal SPM methods will be discussed briefly:

1. As indicated in Fig. 40.5, the images of different spectral signatures have to be registered independently of each other with the smallest amount of cross talk (reciprocal disturbance) possible. On one hand, this requires labeling of t_1, t_2, etc., with fluorochromes of well-distinguishable spectral signatures (e. g., spectrally clearly distinct fluorescence-emission maxima) as well as the possibility of exciting them to fluorescence emission (e.g., using different laser lines and types). An additional important requirement is a sufficient fluorescence photon yield of the fluorochromes used.
2. Detector systems are required that are capable of discriminating the different spectral signatures (e.g., by appropriate optical filters) in a spatially highly resolved way (the FWHM required depending on the problem to be solved); because in nuclear images the fluorescence photon statistics usually is far below optimum, the quantum efficiency and the signal-to-noise relationship of the detector system has to be as good as possible.
3. The precision of localization of the diffraction-image maxima needed for SPM super resolution has to be much better than the

voxel size. For example, a typical voxel size (object space scale) in confocal microscopy is $100 \times 100 \times 250\,\text{nm}$. This fulfills the Nyquist theorem (ca. 1/2 FWHM lateral/axial) but is not sufficient for the precision needed in SPM (e.g., 10 to 100 nm). This problem can be overcome by appropriate image-evaluation algorithms based on the barycentric calculus already introduced by Moebius [25]. Essentially, in its application to the maximum determination relevant here, it means that for each diffraction pattern of an individual target a fluorescence-intensity gravity center is calculated. In case of symmetrical PSFs, the barycenter positions in each spatial direction x, y, z coincide with the maximum positions of the diffraction patterns. Basic considerations show that in this case under the assumptions of an appropriate photon statistics and detector noise, and for geometrically stable relationships between the voxels, this makes it possible to determine the positions of the diffraction pattern maxima with any precision desired.

4. For the required high precision of fluorescence barycenter localization, fluorescence photon statistics (number of effective photons registered per voxel), detector noise and target noise (unspecific fluorescence emission) play a decisive role. Such effects may not only considerably deteriorate the resolution equivalent, but even may contribute to “limited-diffusion” phenomena. For example, computer simulations (Bornfleth et al. unpublished results) indicated that under the assumption of a small number of “effective” photons per voxel typical for observations of small fluorescence labeled targets in living cells, such “limited-diffusion” artifacts may be as large as several hundred nanometer (mean squared distance obtained after 12 evaluation steps) under the boundary conditions of standard confocal microscopy. For an optimal solution of the problem, it is also important to define the voxels assigned to the target image. This may be done by appropriate segmentation algorithms with variable thresholding; see [21] and Chapter 41.

5. In all cases where the different spectral signatures are realized by differences in the fluorescence-emission spectra, a very significant source of error is given by chromatic aberrations, especially in axial direction [26]. Measurements using multispectral calibration beads indicated that depending on the optical microconditions in confocal microscopy, axial chromatic shifts up to about 200 nm were obtained, whereas the lateral chromatic shifts were usually considerably smaller. The contribution of optical microconditions may involve, for example, refraction index variations between microscope front lens and target and axial distance of the target relative to the microscope front lens as well as spectral absorption characteristics

of the medium. Therefore, in situ calibration methods are required to more accurately determine 3-D chromatic shifts [27].

6. In addition to chromatic aberrations, the correction of monochromatic deviations may be important. Even high-performance microscope lenses do not have a completely linear field of view. Thus, the same physical distance of two fluorescing point objects located in the object plane may result in spurious distance variations even under ideal signal-to-noise conditions. For a commercial high-aperture microscope lens, lateral distance variations greater than 50 nm have been measured (Heintzmann, unpublished results). In axial direction, monochromatic distance variations also depend heavily on the optical microconditions. Under conditions relevant for nuclear structure analysis, such aberrations may be as large as 100 to several hundred nanometers (for example, compare [28]). By careful monochromatic calibration, it is experimentally possible to control monochromatic distance errors [21, 26].
7. In real optical systems, the PSF may show deviations from an ideally symmetrical curve. Thus, the calculated barycenter coordinate may not coincide exactly with the coordinate of the maximum (coordinate of the ideal image point). As the final goal is to measure small distances, that is, coordinate differences, in many cases such an error may be eliminated if the PSF has the same kind of deviation for the two targets located at slightly different positions in the object space.
8. Confocal laser scanning microscopes in some way or other use mechanical scanning devices. Thus, mechanical stability and tolerances may influence the correct voxel positioning that is of great importance for the precise subvoxel localization of fluorescence-intensity barycenters necessary for the SPM approach.

Figure 40.4b presents a computer visualization of the capacity of SPM to allow a structural resolution far below the FWHM. On the upper left, as in Fig. 40.4a, a simulated chromatin sequence and its 2-D projections at different rotation angles (first row from above) are shown. The second row then presents the 2-D projections after convolution with a 250-nm PSF. Here, however, it was assumed that each piece of 10-kbp length was labeled with a different spectral signature. The visualization of the spectrally differentiated projections clearly suggests that these images contain structural information below the normal limit of resolution. In the third row, the barycenters of the different spectral intensities were calculated according to the SPM method described in the foregoing. In the end, this simulation example indicates that the SPM method should allow the topological analysis of relatively small nucleosome configurations.

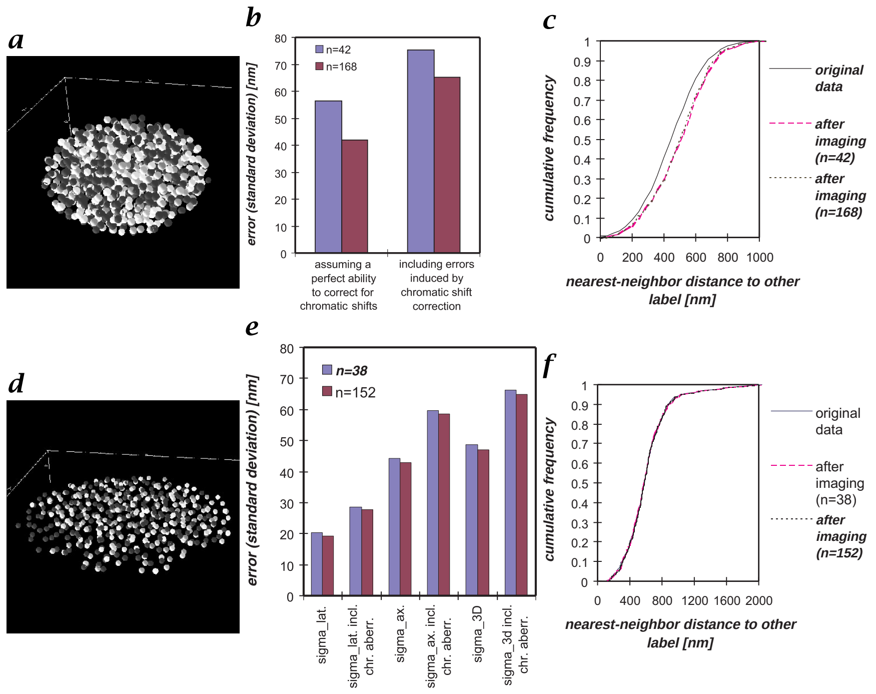


Figure 40.6: ***a*** *Simulation dataset A; 500 FITC spots (i. e., spots carrying a green spectral signature) and 500 CY5 spots (dark-red spectral signature) were randomly distributed in a simulated nuclear volume with radii of 5.2 µm (xy) and 3.1 µm (z). The non-overlapping spots had a diameter of 440 nm;* ***b*** *the standard deviation of nearest-neighbor distances (FITC to CY5) determined for dataset A from the true distances. The columns show the errors in distance determination with and without the ability to correct exactly for chromatic shifts;* ***c*** *the distribution of nearest-neighbor distances between FITC and CY5 in dataset A. The systematic shift towards distances that are too high is due to the optical merging of spots that are close to their neighbors in the same channel;* ***d*** *simulation dataset B. 250 FITC spots and 250 CY5 spots were distributed in a nuclear volume with radii of 9.6 µm (xy) and 1.0 µm (z). The minimum distance between spots was set to 1050 nm;* ***e*** *standard deviations of nearest-neighbor distances (FITC to CY5) determined for dataset B from the true distances. As all spots were found by the algorithm, the standard deviations give a true account of the accuracies possible in lateral and axial direction;* ***f*** *the nearest-neighbor distance distribution of dataset B. A systematic shift is not detectable. (Reprinted from [21] with kind permission of the Royal Microscopical Society.)*

As a second, more realistic example, Fig. 40.6 shows a computer simulation of the accuracy of distance measurements between differentially labeled targets in two-color confocal laser scanning microscopy considering experimental data for photon noise and error of chromatic shift determination [21]. Here, either a completely random distribution of small targets with different spectral signatures within a nuclear volume was assumed (Fig. 40.6a-c), or a random distribution was assumed with the boundary condition that the minimum distance between any two pairs of the same spectral signature was at least one FWHM (Fig. 40.6d-f). Under the assumptions used, the results of the simulation indicated a standard deviation of 3-D distance determination of about 50 nm, that is, about one-tenth the effective FWHM; 3-D distances far below the optical resolution can be determined correctly between targets of different spectral signature.

In analogous experimental high-precision measurements between differentially labeled fluorescent targets, quartz glass beads of (mean ± standard deviation (SD)) 1052 ± 19 nm with a green-yellow fluorescing FITC core of 400 nm (target type I) were mixed with quartz glass beads of 408 ± 22 nm with a red fluorescing RITC core of 200 nm in diameter (target type II), and attached to a cover slip. Besides clusters of beads of type I (FITC-fluorescence), clusters of beads of type II (RITC-fluorescence) were obtained as well as pairs of a type-I bead and a type-II bead adjacent to each other. Three-dimensional imaging of FITC-fluorescence ($specs_1$) and RITC-fluorescence ($specs_2$) was performed with a commercial confocal laser scanning microscope. The axial chromatic shift (lateral chromatic shift was negligible compared to the axial one) was measured using the two clusters of type-I and type-II beads, respectively; the value obtained (225 ± 40 nm) was used to perform the SPM-distance determinations for the barycenters of the type-I type-II-pairs. For the experiment cited here, the mean 3-D distance value determined by SPM was 732 ± 27 nm [29]; the value expected was 734 ± 15 nm. More detailed SPM measurement using the same types of quartz-glass beads with cores of different spectral signature [21] indicated that in the range of 30 to 50 effective photons per voxel (representing a biologically relevant photon statistics), the lateral distance error (SD) was around 20 nm whereas the axial error was between 50 to 75 nm.

40.3 Determination of the resolution equivalent in situ

The accuracy of the SPM method and thus the useful "resolution equivalent" (RE) is given by the capacity to control systematic errors such as chromatic and monochromatic aberrations and to control statistical errors such as deviations in positioning due to photon statistics, detector noise, mechanical instabilities, unknown deviations in the optical microconditions, etc. Besides the mechanical and optical parameters of the recording instruments, sophisticated computer-imaging algorithms are of fundamental importance to obtain an optimal RE. For a direct in situ determination of the RE, let us in a thought experiment assume a cell nucleus containing one "point-like" target t_{cal} (for calibration); this target is simultaneously labeled with two spectral signatures $specs_1$ (e.g., "green"), $specs_2$ (e.g., "red") in such a way that the barycenter coordinates of the fluorescence of $specs_1$ and $specs_2$ in the object space are identical, that is, "true" distances are much lower than the resolution equivalent envisaged. Experimentally, this can be realized with sufficient accuracy by molecular methods of fluorescence labeling, for example, by multicolor FISH of the same genomic target sequence.

By spectrally discriminated registration with an optical system, for example, a multichannel confocal laser scanning microscope, 3-D image stacks for the intensities of $specs_1$ and $specs_2$ are obtained in the image plane. If a system with negligible chromatic and monochromatic shifts is used (or such deviations have been corrected for by independent calibration measurements), the maxima of the image intensity curves I_1 for $specs_1$ and I_2 for $specs_2$ (and hence their fluorescence barycenters) should coincide. Due to all kinds of errors (see the foregoing), however, the experimentally determined maxima (barycenter) positions for the intensity distributions I_1 and I_2 will be found at apparently different locations and hence indicate a spurious distance that does not exist in the object space. Multiple measurements of this kind will eventually give a frequency distribution curve where the abscissa gives the (spurious) distance measured and the ordinate the frequency of cases where this spurious distance was observed. Such a distance-error frequency curve ("resolution equivalent (RE) curve") will give a direct measure of the RE in situ, that is, under the relevant biological conditions to be studied.

As an upper estimate of RE, we may take the full width at half maximum of the distance-error curve (HW-DEC).

Figure 40.7 shows the result of a computer simulation of RE curves assuming confocal 3-D imaging and a number of 6 and 30 effective photons per voxel, respectively. The results indicate that using sufficient photon statistics (realizable in biological labeling experiments) allow 3-D REs much smaller than 100 nm. The HW-DEC using a maximum of 30 photons/per voxel was 15 nm lateral and 45 nm axial. That gives a 3-D RE of about 50 nm.

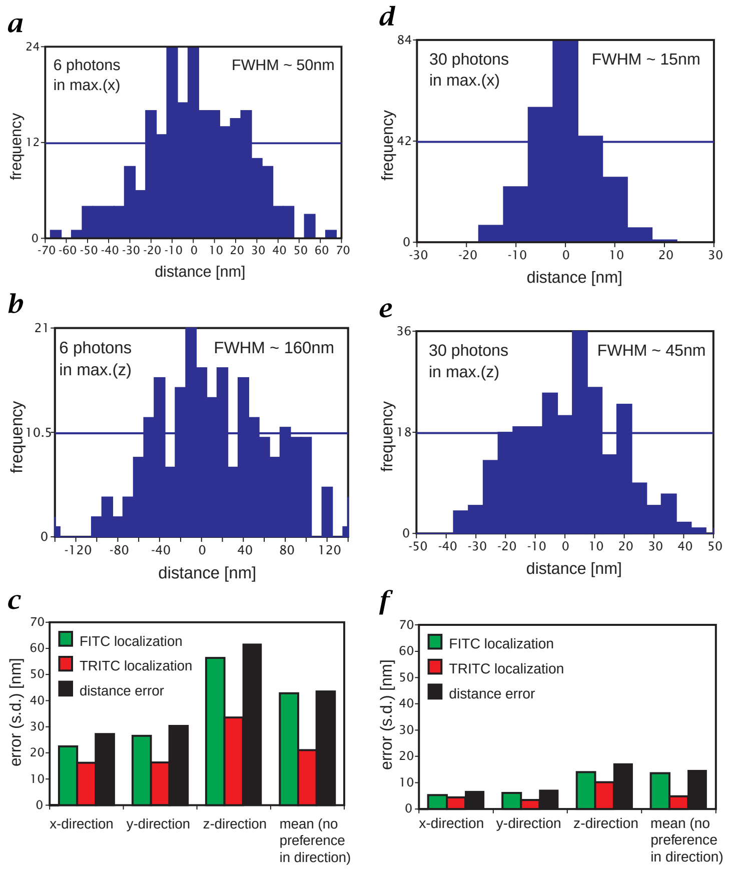


Figure 40.7: *Computer simulation of the RE in situ; 250 simulated targets of 160-nm diameter each were convoluted with a measured PSF of CLSM for a fluorochrome with "green" spectral signature (FITC) and with "red" spectral signature (TRITC), and noise was added. For each spot in the resulting FITC-image, the apparent distance to the nearest neighbor in the TRITC-image was computed after chromatic shift correction. The volume of a voxel was 80 × 80 × 250 nm^3;* ***a*** *For the frequency of measured displacements between FITC- and TRITC- signals (distance-error frequency curve = RE-DEC curve) in x-direction assuming six photons in a voxel of maximum intensity, the values are distributed around the true displacement zero. The full width at half maximum of the distance-error frequency curve (HW-DEC)$_x$ is 50 nm;* ***b*** *the same as in* ***a*** *for the z-direction. The (HW-DEC)$_z$ is 160 nm;* ***c*** *error of localization and distance measurement (standard deviation) for FITC- and TRITC-images (6 photons in a voxel of maximum intensity);* ***d*** *the same dataset as in* ***a,b,c*** *was used, assuming 30 photons in a voxel of maximum intensity. A (HW-DEC)$_x$ of 15 nm was obtained for the x-direction;* ***e*** *the same as in* ***d*** *for the z-direction. The (HW-DEC)$_z$ is 45 nm;* ***f*** *error of localization and distance measurement (standard deviation) for FITC- and TRITC-images (30 photons in a voxel of maximum intensity).*

40.4 Conclusions

Spectral precision distance microscopy (SPM) has opened the way towards an ultrastructural fluorescence far-field microscopy beyond the normal limits of high-resolution microscopy. However, this was only possible because powerful computer hardware and sophisticated image analysis algorithms were available. Here, we highlighted the strong correlation between high-precision optics and high-end computing. Recent applications in 3-D genome pathology have demonstrated the usefulness of this approach using commercially available microscopes. Calibrating and correcting the major parameters discussed in the foregoing, which influence the localization accuracy and distance measurements, confocal laser scanning microscopy with lenses of high numerical aperture revealed a 3-D resolution equivalent of about 50 nm. This offered a first glimpse into leukemia-relevant genome nanostructures on chromosome 9 and 22 (bcr-abl region; A. Esa, personal communication) or chromosome 15 associated structures correlated to the Prader-Willi/Angelmann Syndrom (J. Rauch, personal communication). Moreover, applying SPM to the recently developed SME microscope [19] appears to make a resolution equivalent of less than 10 nm feasible because a localization accuracy of 1 to 2 nm was reported [18]. Finally, it is anticipated that the general principle of SPM, that is, highly accurate localization of spectrally distinguishable targets in combination with sophisticated distance calibration and computer analysis, can be applied to any optical micro- and macro-imaging technique.

Acknowledgments

The authors gratefully acknowledge the financial support of the Deutsche Forschungsgemeinschaft, the German Bundesminister for Education, Science, Research and Technology (Human Genome Project) and the European Union (Biomed-Programme). For stimulating discussions the authors thank A. Esa, J. Rauch, Dr. L. Trakhtenbrot, Dr. Ch. Münkel, Prof. Dr. J. Langowski, and Prof. Dr. T. Cremer.

40.5 References

[1] Kreth, G., Muenkel, C., Langowski, J., Cremer, T., and Cremer, C., (1998). Chromatin structure and chromosome aberrations: modeling of damage induced by isotropic and localized irradiation. *Mut. Res. (in press).*

[2] Cremer, T., Kurz, A., Zirbel, R., Dietzel, S., Rinke, B., Schroeck, E., Speicher, M. R., Mathieu, U., Jauch, A., Emmerich, P., Scherthan, H., Ried, T., Cremer, C., and Lichter, P., (1993). Role of chromosome territories in the functional compartmentalization of the cell nucleus. *Cold Spring Harb. Symp. Quant. Biol.*, **58**:777–792.

[3] Comings, D. E., (1968). The rationale for an ordered arrangement of chromatin in the interphase nucleus. *Am. J. Hum. Genet.*, **20**:440–460.

[4] Cremer, C., Muenkel, C., Granzow, M., Jauch, A., Dietzel, S., Eils, R., Guan, X. Y., Meltzer, P. S., Trent, J. M., Langowski, J., and Cremer, T., (1996b). Nuclear architecture and the induction of chromosomal aberrations. *Mut. Res.*, **366**:97–116.

[5] Muenkel, C., Eils, R., Imhoff, J., Dietzel, S., Cremer, C., and Cremer, T., (1995). Simulation of the distribution of chromosome territories in cell nuclei under topological constraints. *Bioimaging*, **3**:108–120.

[6] Sachs, R. K., van den Engh, G., Trask, B. J., Yokota, H., and Hearst, J. E., (1995). A random-walk/giant loop model for interphase chromosomes. *Proc. Natl. Acad. Sci. USA*, **92**:2710–2714.

[7] Lichter, P. and Cremer, T., (1992). Chromosome analysis by non-isotopic in situ hybridization. In *Human Cytogenetics—a Practical Approach*, D. Rooney and B. Czepulkowski, eds., pp. 157–192. Oxford: IRL-Press.

[8] Eils, R., Dietzel, S., Bertin, E., Schroeck, E., Speicher, M. R., Ried, T., Robert-Nicaud, M., Cremer, C., and Cremer, T., (1996). Three-dimensional reconstruction of painted human interphase chromosomes. Active and inactive X chromosome territories have similar volumes but differ in shape and surface structure. *Jour. Cell Biol.*, **135**:1427–1440.

[9] Rinke, B., Bischoff, A., Meffert, M. C., Scharschmidt, R., Hausmann, M., Stelzer, E. H. K., Cremer, T., and Cremer, C., (1995). Volume ratios of painted chromosome territories 5, 7, and X in female human cell nuclei studied with laser confocal microscopy and the Cavalieri estimator. *Bioimaging*, **3**:1–11.

[10] Muenkel, C. and Langowski, J., (1998). Chromosome structure predicted by a polymer model. *Phys. Rev. E*, **57**:5888–5896.

[11] Zirbel, R. M., Mathieu, U., Kurz, A., Cremer, T., and Lichter, P., (1993). Evidence for a nuclear compartment of transcription and splicing located at chromosome domain boundaries. *Chromos. Res.*, **1**:92–106.

[12] Stelzer, E. H. K., (1998). Contrast, resolution, pixelation, dynamic range and signal-to-noise ratio: fundamental limits to resolution in fluorescence light microscopy. *J. Microsc.*, **189**:15–24.

[13] Rinke, B., Bradl, J., Schneider, B., Durm, M., Hausmann, M., Ludwig, H., and Cremer, C., (1996a). in situ estimates of the spatial resolution for practical fluorescence microscopy of cell nuclei. In *Fluorescence Microscopy and Fluorescent Probes*, S. J, ed., pp. 169–173. New York Washington Boston: Plenum Press.

[14] Bradl, J., Rinke, B., Schneider, B., Edelmann, P., Krieger, H., Hausmann, M., and Cremer, C., (1996a). Resolution improvement in 3D-microscopy by object tilting. *Microsc. Anal.*, **44(11)**:9–11.

[15] Cremer, C. and Cremer, T., (1978). Considerations on a laser-scanning-microscope with high resolution and depth of field. *Microsc. Acta*, **81**: 31–44.

[16] Hell, S., Lindek, S., Cremer, C., and Stelzer, E. H. K., (1994). Fundamental improvement of resolution by 4Pi-confocal microscopy. *Appl. Phys., Lett.*, **64**:1335–1337.

[17] Bailey, B., Farkas, D. L., Taylor, D. L., and Lanni, F., (1993). Enhancement of axial resolution in fluorescence microscopy by standing-wave excitation. *Nature*, **366**:44–48.

[18] Hausmann, M., Schneider, B., Bradl, J., and Cremer, C., (1997). High-precision distance microscopy of 3D-nanostructures by a spatially modulated excitation fluorescence microscope. *Proc. SPIE*, **3197**:217–222.

[19] Schneider, B., Bradl, J., Kirsten, I., Hausmann, M., and Cremer, C., (1998). High precision localization of fluorescent targets in the nanometer range by spatially modulated excitation fluorescence microscopy. In *Fluorescence Microscopy and Fluorescent Probes*, J. Slavik, ed., Vol. 2, pp. 71–76. New York: Plenum Press.

[20] Burns, D. H., Callis, J. B., Christian, G. D., and Davidson, E. R., (1985). Strategies for attaining superresolution using spectroscopic data as constraints. *Appl. Opt.*, **24(2)**:154–161.

[21] Bornfleth, H., Saetzler, K., Eils, R., and Cremer, C., (1998). High-precision distance measurements and volume-conserving segmentation of objects near and below the resolution limit in three-dimensional confocal fluorescence microscopy. *Jour. Microsc.*, **189**:118–136.

[22] Cremer, C., Hausmann, M., Bradl, J., and Rinke, B., (1996a). Verfahren zur multispektralen Präzisionsdistanzmessung in biologischen Mikroobjekten. *Vorläufige Patentanmeldung, Deutsches Patentamt.*

[23] Cremer, C., Edelmann, P., Esa, A., Bornfleth, H., Schneider, B., Bradl, J., Rinke, B., Trakhtenbrot, L., Dietzel, S., Hausmann, M., and Cremer, T., (1998). Spektrale Präzisionsdistanzmikroskopie in der Genomforschung. *Z. Med. Phys., submitted.*

[24] Bradl, J., Rinke, B., Esa, A., Edelmann, P., Krieger, H., Schneider, B., Hausmann, M., and Cremer, C., (1996b). Comparative study of three-dimensional localization accuracy in conventional, confocal-laser-scanning and axial tomographic fluorescence light microscopy. *Proc. SPIE*, **2926**:201–206.

[25] Moebius, A. E., (1827). Der barycentrische Calcul—ein neues Hülfsmittel. In *Analytische Behandlung der Geometrie*. Leipzig: J.A. Barth Verlag.

[26] Manders, E. M. M., (1997). Chromatic shift in multicolour confocal microscopy. *Jour. Microsc.*, **185**:321–328.

[27] Edelmann, P., Esa, A., Hausmann, M., and Cremer, C., (1998). In situ determination of the confocal point-spread function and the chromatic shifts in intact cell nuclei. *Optik (submitted).*

[28] Rinke, B., Bradl, J., Edelmann, P., Schneider, B., Hausmann, M., and Cremer, C., (1996b). Image acquisition and calibration methods in quantitative confocal laser scanning microscopy. *Proc. SPIE*, **2926**:190–199.

[29] Dietzel, S., Eils, R., Saetzler, K., Bornfleth, H., Jauch, A., Cremer, C., and Cremer, T., (1998). Evidence against a looped structure of the inactive human X-chromosome territory. *Exp. Cell. Res., in press.*

41 Three-dimensional Analysis of Genome Topology

Harald Bornfleth[1,2], Peter Edelmann[1,2], Daniele Zink[3], and Christoph Cremer[1,2]

[1]Institut für Angewandte Physik, Universität Heidelberg
[2]Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg
[3]Institut für Anthropologie und Humangenetik, Universität München

41.1 Introduction

The advances in light microscopy that have been achieved in recent years have allowed biologists to obtain 3-D images of cell nuclei via the "optical sectioning" technique. Using this technique, which is offered by confocal laser scanning microscopy (CLSM), a stack of image sections is recorded by scanning a focused laser beam through a 3-D specimen without destruction of the objects under observation (Chapter 21.4). Alongside these developments in microscopy, recent improvements in fluorescent staining techniques have enabled the differential visualization of whole human chromosomes (chromosome "painting" [1], and

Handbook of Computer Vision and Applications
Volume 3
Systems and Applications

ISBN 0–12–379773-X/$30.00

chromosomal subregions, such as early-replicating and late-replicating DNA [2]. The combination of both techniques now allows analysis of relationships between structural and functional features in 3-D conserved interphase cell nuclei, which has become an important issue in biomedical research (see also Chapter 12).

The analysis of confocal microscopic images of structural details in the cell nucleus can contribute greatly to an understanding of relationships between chromosome topology and function. However, the optical sectioning capability of CLSM is limited by the optical resolution in axial direction, which is given by the full width at half maximum (FWHM) of the point-spread function (PSF) (approximately 750 nm under biologically relevant conditions). Nevertheless, it is desirable to obtain information about smaller structural units in the chromosome. An important example for such biological systems are replication foci, that is, domains of DNA that are replicated before the division of the cell, and are labeled during the replication process. Such foci have diameters of approximately 400 to 800 nm [3, 4] and represent structural features that are conserved during the cell cycle. Interestingly, gene-rich foci are replicated earlier than gene-poor foci. Because early-replicating foci hold the vast majority of the active genes, investigating the distribution of the different types of foci inside chromosome territories yields important information about the correlation between functional properties and the topology of the genome.

41.1.1 Overview of methodologies

For the analysis of morphological or positional parameters of fluorescently stained objects in 3-D images of cell nuclei, different types of segmentation approaches were taken. The size of the objects analyzed influenced the method of choice. Chromosomes represent large objects compared to the microscopic observation volume (the observation volume is defined as the volume of an ellipsoid, where the radii are given by half the FWHMs of the PSF in the three scanning directions [5]). After lowpass filtering of the images, a global thresholding procedure with subsequent labeling of connected image voxels gave an estimator of the volume of the painted chromosomes (Cavalieri estimator) [6, 7]. A somewhat different approach stems from computational geometry. The application of Voronoi diagrams to image data uses randomly distributed seeds to tessellate the image objects into polyhedra [8] (Chapter 25). As for the Cavalieri estimator, a user-defined threshold is required to separate polyhedra belonging to the object of interest from background. The smooth outer surfaces given by the polyhedra at the object edges enable the calculation of morphological parameters such as the roundness factor (RF) of an object [9]. The Cavalieri estimator is not directly suited for surface estimates because the rough edges of parallelepipeds representing image voxels artificially increase the object surface. A calibration of the factor by means of test objects of known roundness (e.g., spheres) can correct for these systematic errors.

When the object volume is in the range of the observation volume, the gray-value distribution inside the object is no longer homogeneous, but depends strongly on the microscopic PSF. The Voronoi approach is no longer valid. A voxel-based approach that labeled connected components in images of replication foci was developed by Baumann et al. [10]. They used an interactively defined global threshold and the 26 connectivity condition of foreground image voxels to label different connected regions. They also extracted the number of objects and their volumes. However, for images containing many objects of varying size and shape, a global thresholding procedure is not the method of choice. Instead, many applications use the approach of finding local maxima of intensity in the images. The segmentation of the individual objects can then be performed by using the "anti-watershed" procedure. A discussion of watershed algorithms is presented in Volume 2, Section 21.5.4.

The procedure assigns every point in an image to an object. Thus, watershed algorithms are well suited for the segmentation of datasets with closely adjoining objects, for example, in medical applications [11]. However, in images with large background volumes, many segmented voxels will represent background and thus reduce the accuracy of computation of object loci.

The precise localization of objects is a mandatory requirement for computing distance distributions, which provide a very useful tool to characterize datasets with a multitude of signals. Nearest-neighbor distributions between signals of the same type can describe the amount of clustering, the regularity of patterns or randomness [12]. In biological experiments, different fluorochromes are often used to visualize different types of cellular compartments. Here, co-localization of objects described by the distance between signals in different color channels [13] can suggest attachment sites between compartments [14]. For this purpose, the positions of fluorescent objects had to be determined with subvoxel accuracy. An approach described by Manders et al. [15] uses detection of local maxima and defines the largest contour for each object that contains only one maximum of intensity. All voxels inside the volume described by the contour are used for the calculation of the center of intensity (the center of intensity is the analogon to the center of mass in mechanics, replacing mass density by intensity). However, Manders et al. [15] found that for closely neighbored objects, a systematic error in the localization of each object is introduced by the out-of-focus contribution of intensity from the respective neighbor. This bias is illustrated in Fig. 41.1a showing intensity profiles $f(x)$ and $g(x)$ of two point-like signals with a distance d_{real} that are found after the

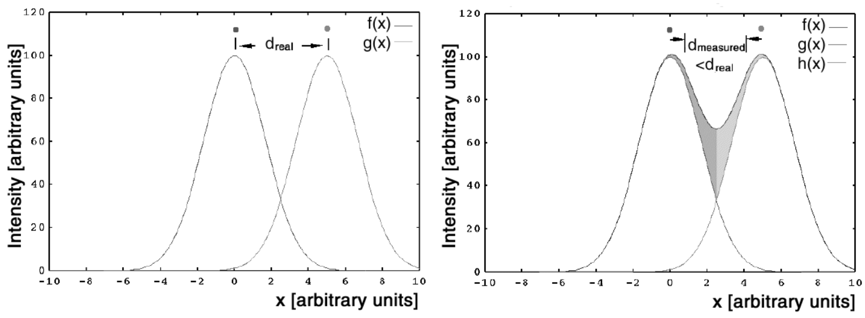


Figure 41.1: Illustration of the systematic bias in distance measurement introduced by neighboring signal contribution (according to Manders et al. [15]).

imaging process. The position of the individual signals can be obtained via the intensity center of mass. The resulting intensity profile $h(x)$ is the enveloping intensity profile (Fig. 41.1b). The gray-shaded areas show the asymmetric components in the intensity profiles. Due to this asymmetry, the intensity centers of mass of the two peaks are shifted towards each other, resulting in a measured distance d_{measured} that is too small.

For the segmentation of objects with a finite size in the range of the microscopic observation volume, a model-based segmentation algorithm was developed by Bornfleth et al. [16]. Briefly, after the detection of local maxima by a top-hat filter shaped to come close to isotropy, a region-growing process was used. From the loci of local maxima, the objects grew along connected shells using the 26-connectivity rule. A voxel connected to the object was recognized as belonging to the object if it fulfilled the fall-set criterion and if its intensity was above an object-specific threshold. The thresholds were adapted in an iterative process (see the following). By setting new individual thresholds in each iteration step, it was possible to match the thresholds according to a calibration function that related the original volume of an object (before the imaging process) to the threshold required for its volume-conserving segmentation. However, the bias in measuring nearest-neighbor distances remained, and computed object volumes were also artificially increased by contributions from neighboring signals.

In this chapter, the performance of morphological shape descriptors for large objects and the accuracy of localization of small objects are investigated. A new powerful shape descriptor is the smoothness factor (SF), a form factor that we introduce in this chapter. It provides a new tool for the shape analysis of objects and shows a very stable behavior in noisy imaging conditions. The performance of the described shape parameters is analyzed using ellipsoids. Their size and shape was matched to that found for chromosome 15 territories. The ellipsoids were subjected to a simulated imaging process including photon shot noise. For spots with a size comparable to the observation volume, we describe an approach that performs the segmentation of each individual spot in an image in a 3-D sub-volume, where the intensity contributions of neighboring spots can be subtracted. Thus, a precise determination of spot volumes even below the microscopic observation volume, and the measurement of spot-spot distances is now possible with enhanced accuracy. The influence of the bias in distance measurement is investigated using model images with randomly distributed spots. These images underwent a simulated imaging process including convolution with the measured microscope PSF of the fluorochromes used, and the application of both additive and multiplicative noise.

The image analysis tools described here were applied to investigate the distribution of replication foci in the chromosome 15 in human fibroblast cell nuclei. The morphology of chromosome 15 was analyzed in metabolically active and inactive cells. A non-random organization of chromatin at the sub-chromosomal level was observed. Apart from the application shown here, the image analysis procedures can be used for a wide variety of applications in morphological and topological analysis of biological structures (e. g., [14, 17]).

41.2 Analysis of large- and small-scale chromatin structure

41.2.1 Morphological parameters of chromosome territories

For the morphological analysis of chromosome territories, the voxel-based representation was used because it corresponds to the experimental setup. After 3-D filtering of the data, the chromosome territories were segmented by interactive setting of a global threshold. After segmentation, the features of interest were extracted by a labeling procedure, which identified all connected voxels of an object using the 26-connectivity rule. To avoid a user-dependent bias in the threshold chosen, the features (chromosome territories) were segmented for a whole range of reasonable thresholds, and for each threshold in this range all the morphological parameters were computed and then averaged. Volumes were computed using the Cavalieri estimator (for definition, see [7]). In addition, the surface area of the chromosome territories and two shape parameters were evaluated:

1. The roundness factor (RF) has previously been used for the analysis of morphological differences between the active and inactive X-chromosome territory in human female amniotic cell nuclei [18]. The dimensionless RF is computed from the volume and the surface

area of the object

$$RF = 36\pi \frac{\text{volume}^2}{\text{surface area}^3} \tag{41.1}$$

The RF yields a value of RF=1 for a perfect sphere with a smooth surface, and lower values for elongated objects and/or objects with a rough surface. However, in some cases this dimensionless ratio between volume and surface area is not dominated by the elongation of the object. Fuzzy boundaries of territories or low signal-to-noise ratios may significantly increase the measured surface area of the territories (see the following), while the computation of the volumes of chromosome territories shows less sensitivity to the fuzziness of their boundary or the signal-to-noise ratio.

2. A new form factor, the smoothness factor (SF) [17], which is very stable with regard to the points mentioned in the foregoing (noise and fuzzy boundaries), was introduced to verify the results of the shape analysis

$$\begin{aligned} SF &= \frac{\int_{\text{sphere}} [(x-x_c)^2 + (y-y_c)^2 + (z-z_c)^2]\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z}{\int_{\text{territory}} [(x-x_c)^2 + (y-y_c)^2 + (z-z_c)^2]\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z} \\ &= \frac{\frac{3}{5}\left[\frac{3}{4\pi}\right]^{\frac{2}{3}} V^{\frac{5}{3}}}{\sum [(x-x_c)^2 + (y-y_c)^2 + (z-z_c)^2]\Delta_x\Delta_y\Delta_z} \end{aligned} \tag{41.2}$$

where (x_c, y_c, z_c) is the center of intensity of the analyzed object; $\Delta_x\Delta_y\Delta_z$ is the volume of one voxel; and V = volume of territory = volume of sphere.

The denominator of Eq. (41.2) gives the mean 3-D distance from the center of intensity of the object of interest to all voxels belonging to the object. The numerator holds the mean 3-D distance for a homogeneous sphere of the same volume as the object analyzed. The ratio yields $SF = 1$ for a sphere, and $SF < 1$ for an elongated object. The second expression is derived by substituting the mean distance from the center of intensity of the sphere to its voxels by its volume. The integral is replaced by a sum over points on a discrete grid.

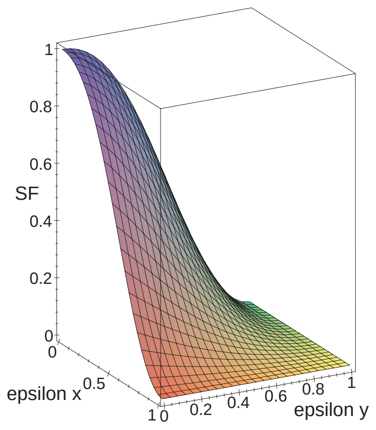


Figure 41.2: *Value of SF for an ellipsoid, where two half-axes are elongated, and the third half-axis is adjusted to maintain the volume of the ellipsoid: Half-axes:* $r_x = R(1 - \epsilon_x),; r_y = R(1 - \epsilon_y),; r_z = R/[(1 - \epsilon_x) * (1 - \epsilon_y)]$.

41.2.2 Topological parameters of subchromosomal targets

A model-based algorithm for the volume-conserving segmentation of signals with volumes comparable to the observation volume of the microscope PSF was described in Bornfleth et al. [16]. The algorithm found local maxima of intensity in 3-D image stacks. These were the centers of fluorescence signals (“spots”). Starting from these spot centers, a region-growing process along connected shells was performed. Each voxel had to meet four criteria to be labeled as belonging to an object:

i. the exclusion criterion stating that the tested voxel had not been previously labeled by another object in the image;
ii. the 26-connectivity criterion;
iii. the fall-set criterion that the voxel intensity was not higher than that of the nearest voxel towards the spot center; and
iv. the threshold-criterion that the voxel intensity was higher or equal to an object-specific threshold.

This threshold depended on the volume that the spot originally had, that is, on the volume of the fluorochrome distribution of the spot. By means of model calculations, a calibration function $T_{\mathrm{rel}}(V)$ was obtained. For each original spot volume V (i. e., the volume before the imaging process), this function gave the corresponding threshold T

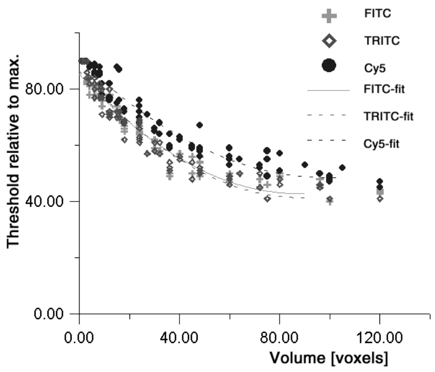


Figure 41.3: *Threshold-volume function found for three different fluorochromes. The segmentation thresholds are approximately constant above a volume of 5.5× the observation volume. Thresholds were obtained from numerical calculations of cube-shaped spots.*

that was required to obtain this original volume after segmentation. However, the original spot volume was not known *a priori* for a given spot. Therefore, the threshold was reduced in each iteration, starting from a threshold that corresponded to a point-like spot. After each iteration, the resulting spot volume was determined. This volume was assumed to be the "true" volume, and the corresponding volume-conserving threshold $T_{\mathrm{rel}}(V)$ for a spot of that size was obtained from the calibration curve. This threshold was used in the next iteration. After 5-10 iterations, a fixed point on the curve was obtained, that is, the volume did not change after application of the new threshold. The function $T_{\mathrm{rel}}(V)$ for the parameters used in the experiments described here is shown in Fig. 41.3. It shows the relative threshold needed for volume-conserving segmentation vs the original spot volume (abscissa) in voxels. The voxel size assumed was $80 \times 80 \times 250\,\mathrm{nm}^3$. The volume-intensity functions were fitted by second-degree polynomials. An earlier version of this plot, showing the results for just two fluorochromes, was published elsewhere [19].

From Fig. 41.3, it is apparent that higher thresholds are required for the segmentation of a spot exhibiting Cy5-fluorescence than for one exhibiting FITC-fluorescence. This is due to the longer wavelength at which Cy5 emits fluorescence light (approximately 670 nm vs 520 nm for FITC). With the small volumes examined, the greater width of the PSF for Cy5 causes more blurring of the signal than that for FITC, that is, the same energy is smeared out over a greater volume for a Cy5 object. This leads to higher thresholds that are needed if the original volume is to be recovered. The model-based segmentation approach was extended to improve the accuracy of volume and distance measurements even in cases where signal overlap lead to an artificial increase in the signal (see Fig. 41.1). After the local maxima in the image had been found, a 3-D voxel cube s_{lmn} was created for each spot[1]. The voxels of the cube were assigned the intensities of the noise-filtered image in the volume range around the spot center (see Fig. 41.4 a), and the cubes were grouped in the new image matrix S_{jlmn}. The gray values in the new, 4-D subvolume matrix S, with the elements $\{0..J = N_{\text{spots}} - 1; 0..L; 0..M; 0..N\}$ were assigned as

$$S_{j,L/2+l,M/2+m,N/2+n} := g'_{z_j+l,y_j+m,x_j+n}, \quad \text{where} \quad \begin{bmatrix} 0 \le j < N_{\text{spots}} \\ -\frac{L}{2} \le l < \frac{L}{2} \\ -\frac{M}{2} \le m < \frac{M}{2} \\ -\frac{N}{2} \le n < \frac{N}{2} \end{bmatrix} \tag{41.3}$$

with the number of spots N_{spots} and the coordinates $\{x_j, y_j, z_j\}$ of the local maximum found for spot j; g'_{lmn} is the image matrix of the preprocessed image. A second 4-D matrix T_{jlmn} contained 3-D subvolumes with the labeled voxels for each spot #j. Initially, only the location of the central voxel, that is, $T_{.,L/2,M/2,N/2}$, was labeled for all spots. Each spot was then allowed to grow inside its cube in shells around the spot center according to criteria (ii) to (iv) described in the foregoing. The segmentation threshold was initially set to 82 % of the maximum intensity of a spot, because this corresponded to the threshold necessary for segmentation of a point-like object. A growth shell was completed for all spots before the next shell was tested to give all spots equal possibilities for expansion. To prevent an overlap between spots in the same color channel, each labeled voxel was marked not only in the label matrix T, but also in the original image. If a voxel tested in a growth shell had already been labeled in the original image as belonging to another spot, it was rejected. Thus it was possible to prevent spots from merging with neighboring spots (see criterion (i)). After the growth process was completed for all spots, the labeled voxels were used for a least-squares fit of spot intensity. According to the assumption of a constant fluorochrome distribution inside the spot, each voxel of a spot labeled in the label matrix T was convoluted with the microscopic median-filtered PSF h_m

$$F_j = T_j * h_m, \quad \text{where} \quad * \quad \text{denotes the 3-D convolution} \tag{41.4}$$

F_j denotes the 3-D subvolume of the 4-D matrix F containing the convoluted intensities for spot #j. In order to reproduce the intensity distribution of the spot #j after imaging, a least-squares fit of the resulting

[1]The subvolume had fixed dimensions in x-, y-, and z-direction. As the number of elements was smaller in z-direction due to the larger pixel size, it is not entirely correct to speak of an image cube. However, this nomenclature is used for simplicity.

model image cube F_j to the original data in S_j was performed according to Brent's method [20] for each spot ($j \in [0..N_{spots} - 1]$). This resulted in F holding the fitted intensities. In a final step, the image cubes S_j were reloaded from the original image, and for each spot #j, the fitted intensities were used to subtract the intensity contributions of neighboring spots in the image cubes of individual spots (see Fig. 41.4 b)

$$F'_{jlmn} = \sum_{\kappa=0}^{N_{spots}-1, \kappa \neq j} \{F_{\kappa\lambda\mu\nu}, \quad \text{if} \quad \left[\begin{array}{c} \lambda := l - (z_\kappa - z_j) \in [0..L] \\ \mu := m - (y_\kappa - y_j) \in [0..M] \\ \nu := n - (x_\kappa - x_j) \in [0..N] \end{array} \right] \} \tag{41.5}$$

$$S'_{jlmn} = S_{jlmn} - F'_{jlmn} \tag{41.6}$$

The condition behind the sum holds the overlap condition: Only if the spacing between the local maxima of two spots is sufficiently small (e. g., the value of $x_\kappa - x_j$ for the x-direction), is a subtraction of intensities performed for the spot $\#\kappa$. Otherwise, intensity contributions from spot $\#\kappa$ are not subtracted from spot # j. The 4-D array S' holds the corrected intensities. Because in the first iteration the spots had not yet acquired their final volume due to the initial high threshold, the accuracy of the fitting process increased only gradually during the iterations, along with the accuracy of the subtraction process. The least-squares fitting was mandatory to obtain a guess about the intensity distribution that resulted from the expected spot geometry. If a fit did not reproduce the experimental data with sufficient accuracy (defined as a value of $\chi^2/n > 30$, where n is the number of voxels used for the fit, and χ^2 is the sum over the squares of the deviations of intensities), the fitted intensities for the respective spot were not subtracted from other spots in that iteration. The resulting intensity distributions in the cubes were used in the next iteration step (see Fig. 41.4 c), where new individual thresholds were defined according to the function T_{rel}(V), based on the volume that a spot had acquired in the last growth step. The process converged after 5-10 iterations. The process of fitting spot intensities after segmentation to the original data allowed the mean brightness of a spot as well as its volume and position to be evaluated. Because the value for the mean brightness was obtained by the fit after convoluting a segmented volume with the PSF, it gave a measure for the fluorochrome density inside the probe, which yielded rather more accurate results than simply adding up intensities in the image.

An example for the reduction of neighboring spot signals by the use of individual subvolumes is given in Fig. 41.5 showing the procedure for an image of touching microspheres with a quartz glass shell

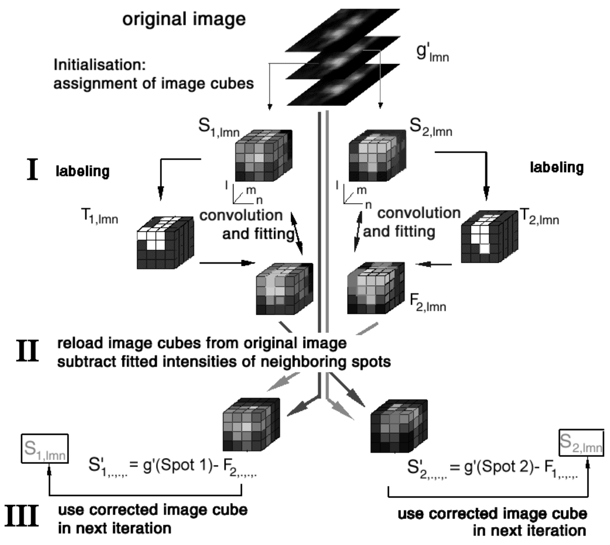

original image
Initialisation: assignment of image cubes
g'lmn
S1,lmn
S2,lmn
I labeling
labeling
T1,lmn
convolution and fitting
convolution and fitting
T2,lmn
F2,lmn
II reload image cubes from original image subtract fitted intensities of neighboring spots
S1,lmn
S'1,...= g'(Spot 1)- F2,...
S'2,...= g'(Spot 2)- F1,...
S2,lmn
III use corrected image cube in next iteration
use corrected image cube in next iteration

Figure 41.4: *Schematic diagram of the steps taken by the subtraction algorithm.*

around a fluorescent core. The quartz glass microspheres had an overall diameter of 416 nm, with a fluorescent core 200 nm in diameter, well below the microscopic observation volume. This placed great demands on the volume-conserving segmentation. A central section from a 3-D data stack is presented. The contour plots show the intensity distributions in the central section surrounding example beads A (left) and B (right). The situation at the beginning (c) and after the subtraction of neighboring intensities ((d), after seven iterations) is shown. Figure 41.5 shows that volumes are recovered after segmentation. For the quartz glass spheres, the gap between cores is clearly visible ((b); different gray shades denote different objects). Using the approach described in [16] without subtraction, the segmented cores touched after segmentation (not shown). The distance between the two example spots A and B was determined to be 359.4 nm with the first algorithm, and 432.1 nm with the subtraction of intensities. Because the manufacturer reported a size of the spheres as 416 ± 22 nm [21], this example demonstrates the bias effect and its reduction using the new approach.

The evaluation time for spot growth and fitting increased linearly with the number of spots, whereas the subtraction time was more closely related to (number of spots)^2 (for the worst-case scenario that all spots had mutual intensity contributions). The total evaluation time was proportional to (number of spot$s)^a, a \in [1, 1.5]$. On a Silicon Graphics workstation (SGI IRIS INDIGO, CPU R4400, 200 MHz, 96 MB

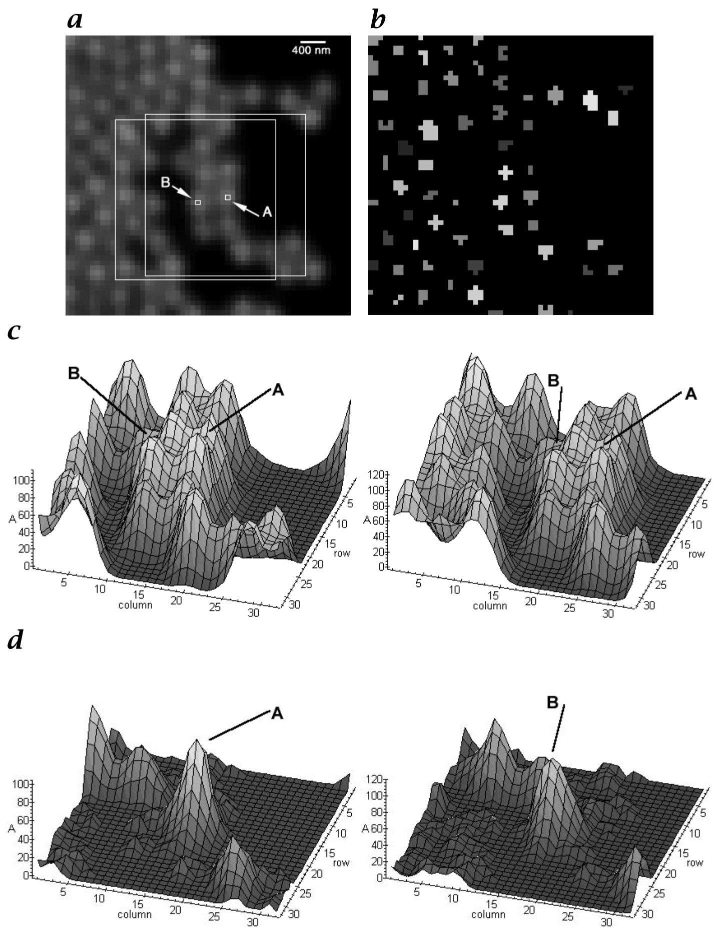


Figure 41.5: *Segmentation of quartz glass microspheres with a total diameter of 416 nm and a fluorescent core of 200-nm diameter. A central section from a 3-D data stack is shown.*

RAM), the segmentation of a model nucleus containing 500 spots required approximately 2 h CPU time, whereas the segmentation of a subset containing 50 spots was completed in around 10 min.

41.2.3 Biological experiments and data acquisition

As an example for the application of the methods presented, human fibroblast cell nuclei where chromosome 15 was visualized by chromosome painting were evaluated. Additionally, early-replicating chromatin and late-replicating chromatin in chromosome 15 was visualized with two different fluorochromes [4, 22]. Series of light-optical sections

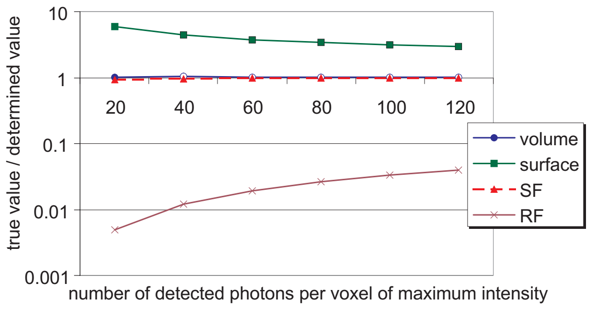
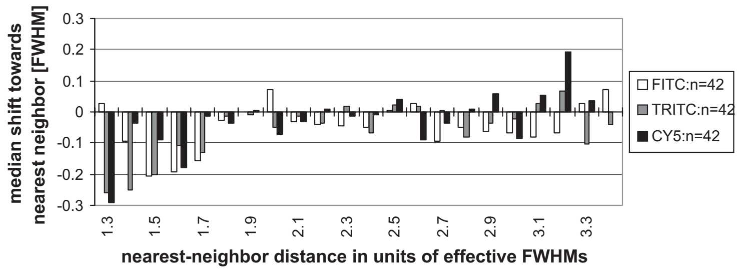


Figure 41.6: *Morphology analysis of simulated chromosome territories. Shape estimators are plotted vs the number of detected photons.*

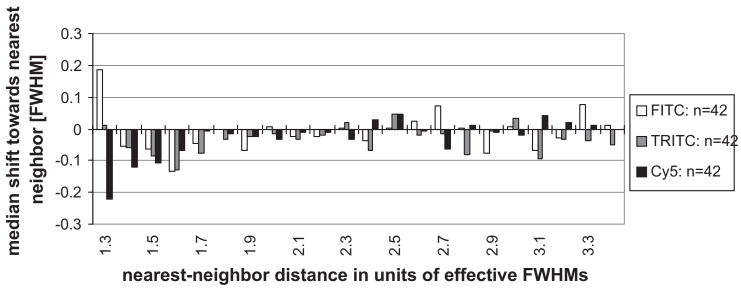
were recorded with a 3-channel Leica TCS 4D confocal laser scanning microscope as described in [18]. A sampling rate of 100 nm/pixel in lateral direction and of 250 nm/section in axial direction was chosen in order to meet the requirements of the Nyquist theorem. To avoid crosstalk between different fluorochromes, each section was scanned sequentially for each individual fluorochrome (first, Cy5, then TRITC, then FITC) before the stage moved on to the next section. The chromatic shift in axial direction between the fluorochromes was determined in separate experiments. It was found to be 230 nm (s.d.=60 nm) between FITC and TRITC, and 8 nm (s.d.=29 nm) between TRITC and Cy5 [16].

41.2.4 Performance tests of morphology analysis

The performance of the morphology analysis was evaluated using Monte Carlo simulations. Ellipsoids of known size and shape served as test models. After introducing additive and multiplicative noise to the model images following the Poisson statistics, the ellipsoids were segmented and all morphological parameters described in the foregoing were computed. Figure 41.6 shows the relative error of the computation of the parameter volume, surface area, roundness factor (RF) and smoothness factor (SF) for different numbers of detected photons in a voxel of maximum intensity. The parameter volume and SF could be determined with high accuracy. Here, the dependence on the photon statistics was negligible. In contrast, the surface area was strongly overestimated by a factor that depended on the given number of detected photons. Therefore, the determination of surface area and roundness factor (RF) was greatly dependent on the number of detected photons.



a



b

Figure 41.7: *Median shift in nearest-neighbor distances observed after segmentation of spots in simulated images. The abscissa denotes the distance of a spot to its nearest neighbor in units of effective FWHMs. The ordinate denotes the median shift between true distance and distance after segmentation that was observed for each distance bin:* ***a*** *median shifts observed with a threshold-based segmentation method that does not subtract neighboring intensities. (Reprinted from Bornfleth et al. [16] with kind permission of the Royal Microscopic Society.);* ***b*** *median shifts observed after subtraction of neighboring intensities. The bias towards measured distances that are too small is significantly reduced.*

41.2.5 Bias reduction in distance measurements

The bias in distance measurements that remained after segmentation of fluorescent objects with the algorithm described previously was investigated using model images. The test dataset used was described in [16]. It consisted of 500 spherical domains (diameter 440 nm) that were randomly distributed in an ellipsoidal volume representing the nucleus. The spheres were allowed to touch, but not to overlap. The resulting image was convoluted with the measured PSFs for FITC, TRITC and Cy5, respectively, and noise was added. A maximum of 41 detected photons was assumed for noise simulation. Figure 41.7 shows the median shift in nearest-neighbor distances that was observed after segmentation of the domains. The distances are given in units of the effective FWHM of the PSF used. This effective FWHM is obtained by scaling the distances in lateral direction with the half-width of the PSF in lateral direction, and by scaling the distances in axial direction with the axial PSF half-width. Because the measured distances are smaller than the real distances for closely neighbored targets, the median shift is negative. Compared to the results from Bornfleth et al. [16], (Fig. 41.7 a), the bias is significantly reduced. It is only marginally higher than the fluctuations of the shift that are due to statistical variations (Fig. 41.7 b). This means that clustering can be investigated using the nearest-neighbor statistics without a significant distortion of the results due to systematic errors. As the spots had a size of 440 nm, distances below 1.3 effective FWHMs, where no correct segmentation was possible, were not evaluated.

41.2.6 Cell-cycle specific morphology

As an example for the application of the shape parameters to biological data, the morphology of chromosome 15 territories was analyzed in two phases of the cell cycle. The parameters computed were the volume of the territories after segmentation, their smoothness (see Eq. (41.2)) and their roundness (defined in Eq. (41.1)). Thirty-one G1 territories and 44 G0 territories were analyzed. Figure 41.8 shows that the volumes of territories did not differ significantly between the G1 and G0 phase (p=0.244, Kolmogorov-Smirnov test). In contrast, the smoothness of chromosome 15 territories showed a strong dependence on the cell cycle: the values were generally smaller in G1. This difference was significant (p=0.003). An even more pronounced difference emerged from the roundness analysis, where the difference was significant at a level of p=0.0004. This implies that the surface of segmented chromosome territories was larger in the G0 stage, where the metabolical activity was small. The departure of G0 chromosome territories from a round shape can be seen as a change of degree of order. This corresponds well with an earlier study that investigated a DNA target on chromosome 11. The authors found that the position of the target was random in G0 phase, but nonrandom in G1 phase [23].

41.2.7 Cluster analysis of early- and late-replicating foci

The spatial distribution of early- and late-replicating foci in chromosome 15 territories was investigated by statistical methods. Figure 41.9 shows the replication foci of a territory in G1 phase after 3-D reconstruction of the segmented image. An MPEG video of the segmented territory is available (`/movies/41` on the CD-ROM). The lines between foci denote the distance vectors to the nearest neighbor. A colored version is given in the color plates. Red and green lines denote distance vectors between foci of the same type, whereas yellow lines show the distance vectors between foci of different types, that is, between early-replicating (red) and late-replicating foci (green). The visualization routine was written in OpenGL.

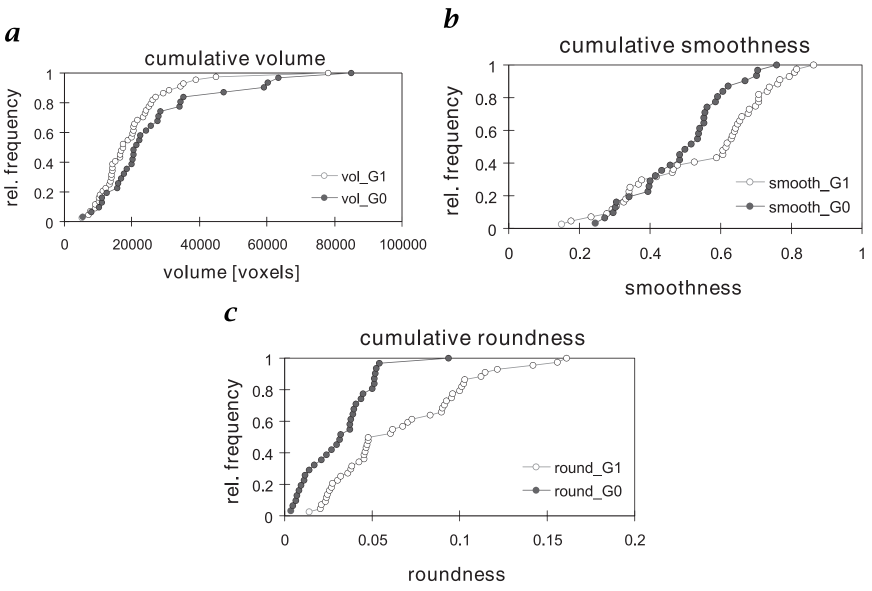


***Figure 41.8:** Morphological parameters computed for chromosome 15 territories in G1 phase and in G0 phase. Cumulative plots are presented: **a** the volumes computed do not show significant differences; **b** the smoothness factors show a significant difference between territories in G1 phase and G0 phase; **c** the difference is even more pronounced using the roundness parameter.*

Figure 41.10 shows the median values of comparative nearest-neighbor distances obtained for 18 territories in G1 cells. For both early- and late-replicating foci, the distance to the nearest neighbor in the other color channel was on average greater than the distance to the neighbor in the original color channel. This means that on average, early-replicating foci were closer to other early-replicating foci than to other late-replicating foci. This is surprising as it is expected that early-replicating foci and late-replicating foci alternate on the condensed metaphase chromosomes. This observed clustering of foci of the same functional state is not due to artifacts introduced by the segmentation algorithm because the algorithm reduces the bias in distance measurements effectively. It suggests that early-replicating foci and late-replicating foci form higher-order clusters in interphase nuclei.

41.3 Discussion and outlook

In this chapter, new image analysis tools to investigate morphological and topological features of chromosomes were introduced. Their performance was tested with regard to statistical and systematic errors. As an example of the usefulness of the methods, they were ap-

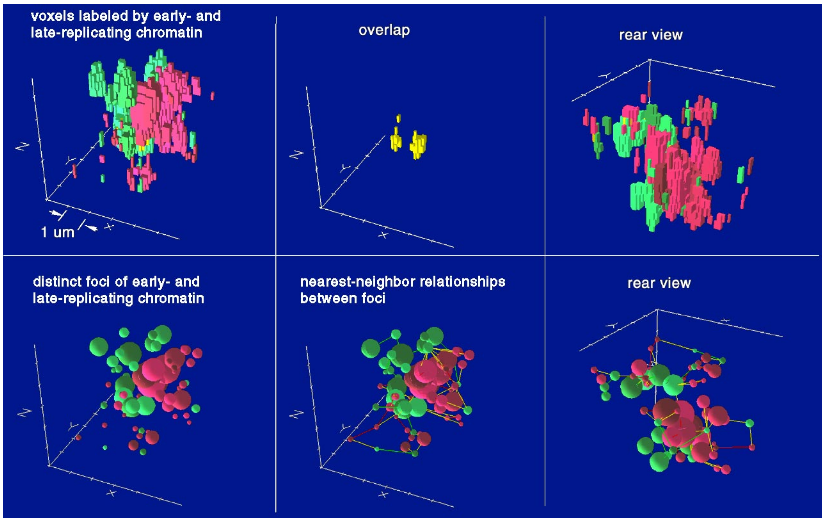

voxels labeled by early- and late-replicating chromatin
overlap
rear view
1 um
distinct foci of early- and late-replicating chromatin
nearest-neighbor relationships between foci
rear view

Figure 41.9: *Visualization of a segmented chromosome 15 territory in G1 phase. Upper row: the segmented voxels for early-replicating chromatin (dark gray) and late-replicating chromatin (light gray). Lower row: the individual foci found by the segmentation algorithm, shown as spheres with a radius that yields the individual volume found. Distance vectors between nearest neighbors are denoted by lines in the middle and on the right for two different perspectives; for a movie see `/movies/41/chrom_15.mov`; (see also Plate 18).*

plied to study the morphology of chromosome territories in different stages of the cell cycle. This analysis revealed a cell-cycle specific shape of chromosome 15 territories. In addition, nearest-neighbor distance measurements provided evidence for a clustering of subchromosomal foci in chromosome territories.

The morphology analysis of simulated chromosome territories revealed that two of the parameters, volume and smoothness factor, can be determined with high accuracy. Furthermore, this high accuracy is not affected by the number of detected photons within a large range. Therefore, the Cavalieri-estimator in combination with the new shape parameter, the smoothness factor (SF), allows the precise analysis of volume and shape of chromosome territories. The computation of the absolute values for surface area and consequently for the third parameter, the roundness factor, did not give the correct results. The surface area was overestimated by a constant factor for each given number of detected photons per voxel. Therefore, we conclude that surface area and roundness factor should only be used to compare different territories, not for the computation of absolute quantities. Care has to be taken that only data recorded under the same photon statistics are compared.

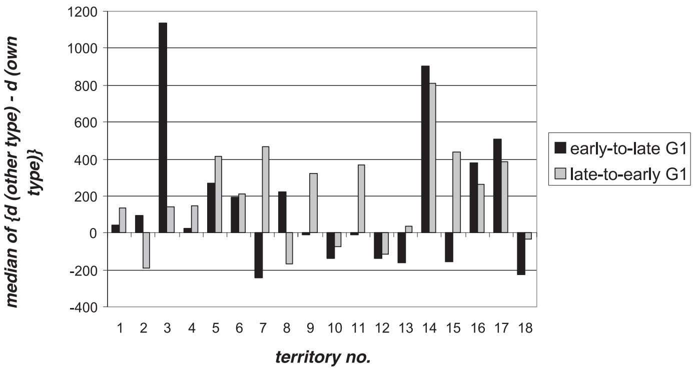


Figure 41.10: *Example for the analysis of comparative nearest-neighbor distances in chromosome 15 territories. For each of 18 territories in G1 phase, the median value of the distance to the nearest neighbor of the other type, minus the distance to the nearest neighbor of the same type, is given. It is apparent that most values are* > 0*, that is, foci are closer to the nearest neighbor of the same type than to the nearest neighbor of the other type.*

Concerning the segmentation and analysis of small targets (i. e., of targets with a size comparable to the microscopic observation volume), artifacts that are due to fundamental limitations of the imaging process play an important role. Especially, distance measurements are affected by an out-of-focus signal from other labeled structures. Because the precise measurement of distances can provide a means to overcome the resolution limit for specialized applications (Chapter 40), it is of utmost importance to correct for systematic errors. The nearest-neighbor analysis presented as an example in this paper also has to rely on unbiased distance measurements. The algorithm presented here enables the subtraction of an out-of-focus signal from neighboring spots in regions where the diffraction-limited signal patterns of two or more spots overlap. It is possible to retain the complete diffraction image for each spot, with the contributions of other diffraction-limited signals subtracted. The whole signal that was created by a spot can be used for the calculation of its center of intensity, which enhances the accuracy of localization. A drawback of the method is that it works with median-filtered images. Some of the accuracy is lost due to nonlinear filtering effects. Future work will concentrate on including both raw and noise-filtered images into the segmentation algorithm.

It could be shown that the new approaches succeeded in finding new quantitative parameters to describe functional topology and morphology of chromosomes and subchromosomal targets. The methods will be applied to further biological experiments in the near future.

Acknowledgments

We are grateful to Rainer Heintzmann for assistance in computer programming, to Dr. Michael Hausmann and Prof. Dr. Thomas Cremer for helpful discussions. This project was supported by the Deutsche Forschungsgemeinschaft, the Federal Minister of Education, Science, Research and Technology (BMBF) and the European Union.

41.4 References

[1] Cremer, T., Lichter, P., Borden, J., Ward, D. C., and Manuelidis, L., (1988). Detection of chromosome aberrations in methaphase and interphase tumor cells by in situ hybridization using chromosome specific library probes. *Hum. Genet.*, **80**:235–246.

[2] Fox, M. H., Arndt-Jovin, D. J., Jovin, T. M., Baumann, P. H., and Robert-Nicoud, M., (1991). Spatial and temporal distribution of DNA replication sites localized by immunofluorescence and confocal microscopy in mouse fibroblasts. *J. Cell Sci.*, **99**:247–253.

[3] Berezney, R., Mortillaro, M. J., Ma, H., Wei, X., and Samarabandu, J., (1995). The nuclear matrix: a structural milieu for genomic function. *Int. Rev. Cytology*, **162A**:1–65.

[4] Zink, D., Cremer, T., Saffrich, R., Fischer, R., Trendelenburg, M., Ansorge, W., and Stelzer, E. H. K., (1998). Structure and dynamics of human interphase chromosome territories in vivo. *Hum. Genet.*, **102**:241–251.

[5] Lindek, S., Cremer, C., and Stelzer, E. H. K., (1996). Confocal theta fluorescence microscopy with annular apertures. *Appl. Optics*, **35**:126–130.

[6] Bischoff, A., Albers, J., Kharboush, I., Stelzer, E. H. K., Cremer, T., and Cremer, C., (1993). Differences of size and shape of active and inactive X-domains in human amniotic fluid cell nuclei. *J. Micr. Res. Techn.*, **25**: 68–77.

[7] Gundersen, H. J. and Jensen, Z. B., (1987). The efficiency of systematic sampling in stereology and its prediction. *J. Microsc.*, **147**:229–263.

[8] Bertin, E., Parazza, F., and Chassery, J. M., (1993). Segmentation and measurement based on 3D voronoi diagram: application to confocal microscopy. *Computerized Medical Imaging and Graphics*, **17**:175–182.

[9] Eils, R., Bertin, E., Saracoglu, K., Rinke, B., Schröck, F., E.and Parazza, Usson, Y., Robert-Nicoud, M., Stelzer, E. H. K., Chassery, J. M., Cremer, T., and Cremer, C., (1995). Application of confocal laser microscopy and three-dimensional Voronoi diagrams for volume and surface estimates of interphase chromosomes. *J. Microsc.*, **177**:150–161.

[10] Baumann, P. H., Schormann, T., and Jovin, T., (1992). Three-dimensional component labeling of digital confocal microscope images enumerates replication centers in BrdUrd labeled fibroblasts. *Cytometry*, **13**:220–229.

[11] Wegner, S., Harms, T., Oswald, H., and Fleck, E., (1996). Medical image segmentation using the watershed transformation on graphs. In *IEEE International Conference on Image Processing*. Lausanne.

[12] Schwarz, H. and Exner, H. E., (1983). The characterization of the arrangement of feature centroids in planes and volumes. *J. Microsc.*, **129**:155–169.

[13] Manders, E. M. M., Verbeek, F. J., and Aten, J. A., (1993). Measurement of co-localization of objects in dual-colour confocal images. *J. Microsc.*, **169**: 375–382.

[14] Eggert, M., Michel, J., Schneider, H., S. Bornfleth, Baniahmad, A., Fackelmayer, F. O., Schmidt, S., and Renkawitz, R., (1997). The glucocorticoid receptor is associated with the RNA ninding nuclear matrix protein hnRNP U. *Jour. Biological Chemistry*, **272**:28471–28478.

[15] Manders, E. M. M., Hoebe, R., Strackee, J., Vossepoel, A. M., and Aten, J. A., (1996). Largest Contour Segmentation: A tool for the localization of spots in confocal images. *Cytometry*, **23**:15–21.

[16] Bornfleth, H., Sätzler, K., Eils, R., and Cremer, C., (1998). High-precision distance measurements and volume-conserving segmentation of objects near and below the resolution limit in three-dimensional confocal microscopy. *J. Microsc.*, **189**:118–136.

[17] Edelmann, P., Bornfleth, H., Dietzel, S., Hausmann, M., Cremer, T., and Cremer, C., (1997). Dreidimensionale Mehrfarben-Bildrekonstruktion zur Untersuchung der Position der Gene ANT2 und ANT3 auf menschlichen X-Chromosomenterritorien. In *10. Heidelberger Zytometrie Symposium, Kurzfassung der Beiträge*, pp. 23, ISSN 0949–5347.

[18] Eils, R., Dietzel, S., Bertin, E., Schröck, E., Speicher, M. R., Ried, T., Robert-Nicoud, M., Cremer, C., and Cremer, T., (1996). Three-dimensional reconstruction of painted human interphase chromosomes: Active and inactive X chromosome territories have similar volumes but differ in shape and surface structure. *J. Cell Biol.*, **135**:1427–1440.

[19] Bornfleth, H., Zink, D., Sätzler, K., Eils, R., and Cremer, C., (1996). Modellgestützte Segmentierung von Replikationsdomänen in dreidimensionalen konfokalen Mikroskopiebildern. In *Mustererkennung 1996*, B. Jähne, P. Geissler, H. Haussecker, and F. Hering, eds., pp. 408–419. Heidelberg New York Tokyo: Springer.

[20] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B., (1992). *Numerical Recipes in C: The Art of Scientific Computing. Second edition.* Cambridge New York: Cambridge University Press.

[21] Verhaegh, N. A. M. and van Blaaderen, A., (1994). Dispersions of rhodamine-characterization, and fluorescence confocal scanning laser microscopy. *Langmuir*, **10**:1427–1438.

[22] Zink, D., Bornfleth, H., Visser, A., Cremer, C., and Cremer, T., (1998b). Interphase chromosome banding: Approaching the functional organization of human chromosome territories. *Exp. Cell Res. (Manuscript submitted).*

[23] Hulspas, R., Houtsmuller, A., Krijtenburg, P., Bauman, J., and Nanninga, N., (1994). The nuclear position of pericentromeric DNA of chromosome 11 appears to be random in G0 and non-random in G1 human lymphocytes. *Chromosoma*, **103**:286–292.

Index

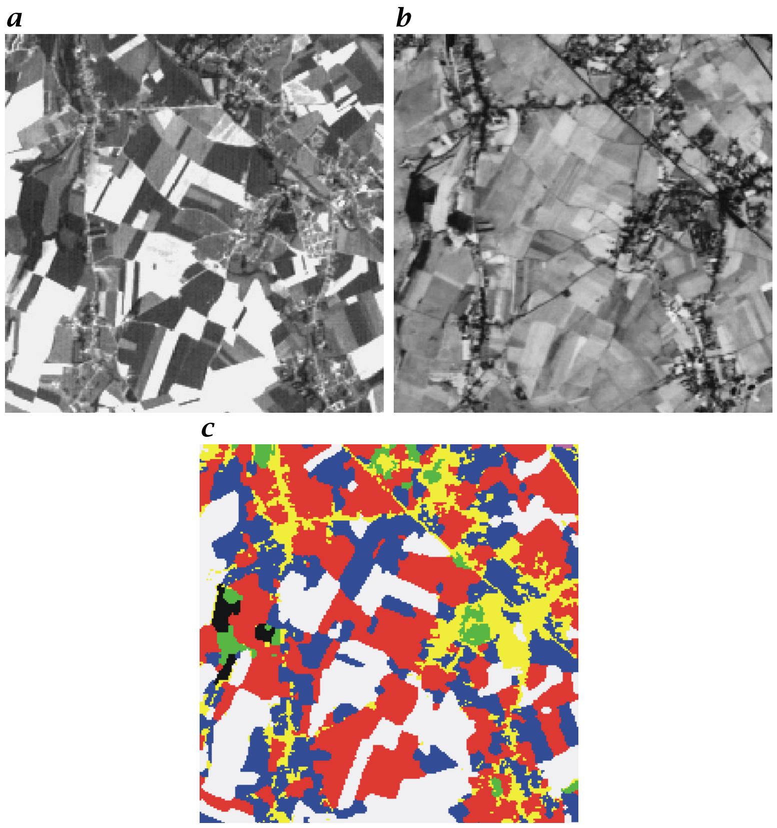


Plate 1: *Classification of a multispectral satellite image (the two input bands are shown as grayscale images). The result of the classification is shown as a labeled image, a specific color being used for each ground cover (e. g., yellow for urban areas and green for forests); (see also Fig. 12.1, p. 285)*

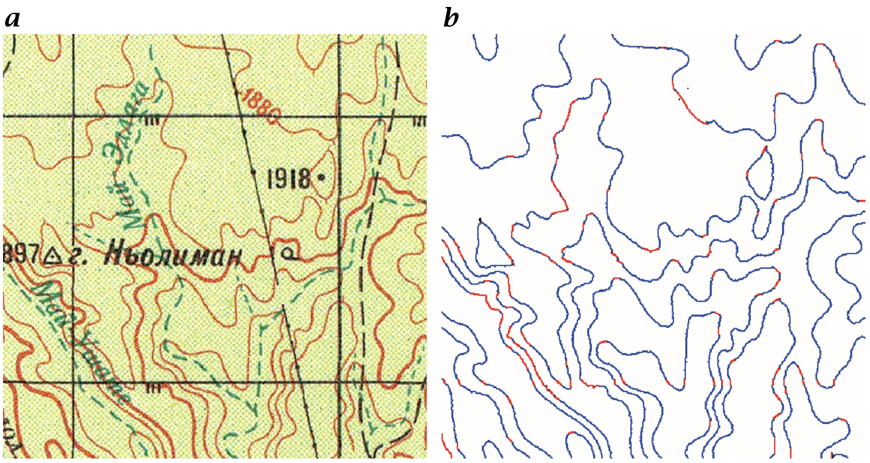


Plate 2: ***a*** *Digitized topographic map;* ***b*** *extracted contour lines (connections of initially disconnected lines are shown in red); (see also Fig. 19.1, p. 412)*

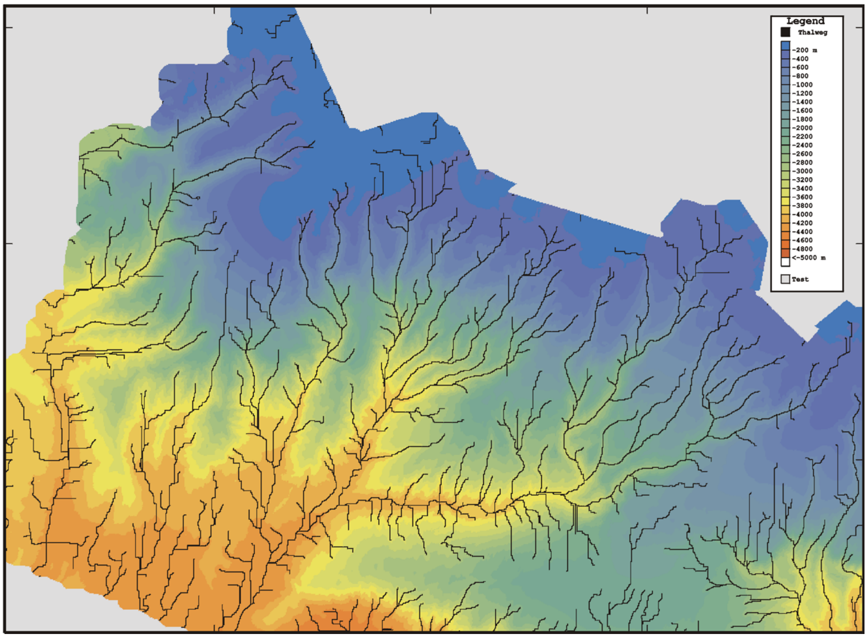


Plate 3: *Submarine drainage networks extracted from bathymetric data of the approaches of the Channel in the NE Atlantic. The cell size is* 200 *m* × 200 *m. The network has been extracted using a threshold of 300 cells for the cumulative drainage area. The canyon network incises the continental slope from the shelf (-200 m) to the abyssal plain (-4500 m); (see also Fig. 19.16, p. 425)*

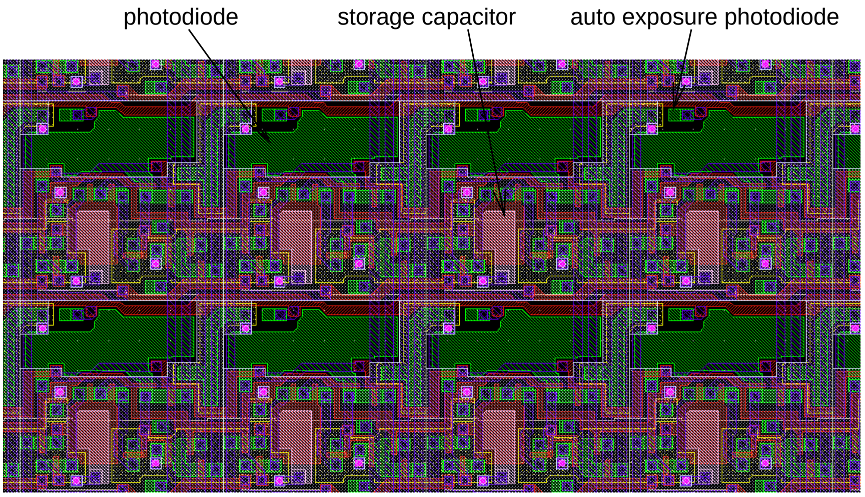


Plate 4: *Layout of eight pixels in the self-calibrating vision chip. Image-acquisition diodes, exposure-control diodes and storage capacitors are indicated; (see also Fig. 26.3, p. 535)*

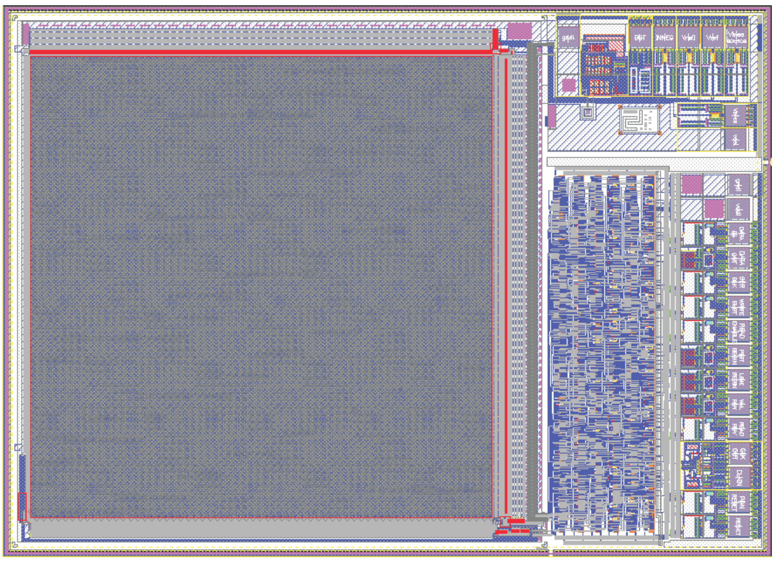


Plate 5: *Layout of the complete vision chip. The digital control part made of standard cells and the quadratic pixel matrix with decoder and amplifier arrays can be seen; (see also Fig. 26.4, p. 536)*

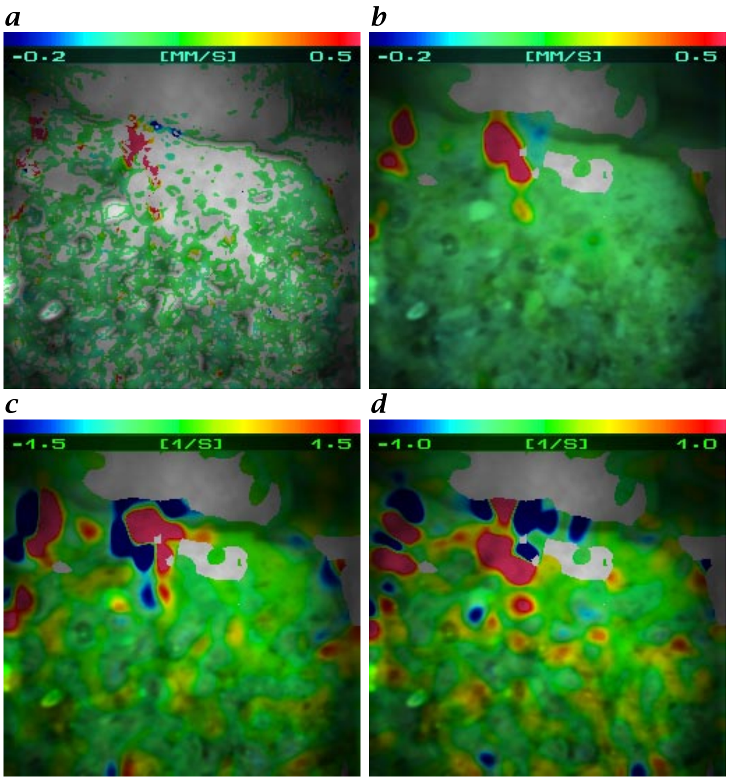
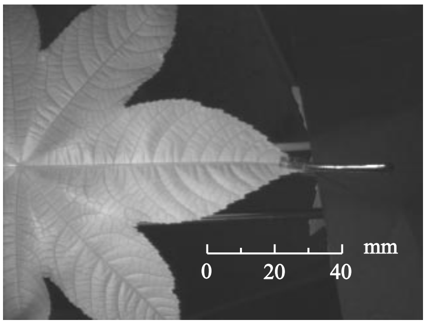
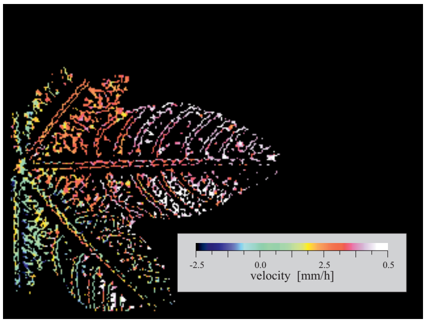


Plate 6: *Interpolation of a dense displacement vector field from a sparse field by normalized convolution and calculated characteristics of the velocity field:* ***a*** *sparse vertical velocity;* ***b*** *interpolated velocity;* ***c*** *value of the rotation; and* ***d*** *divergence; (see also Fig. 32.9, p. 709)*

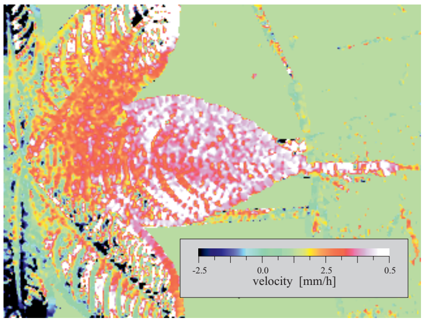
a



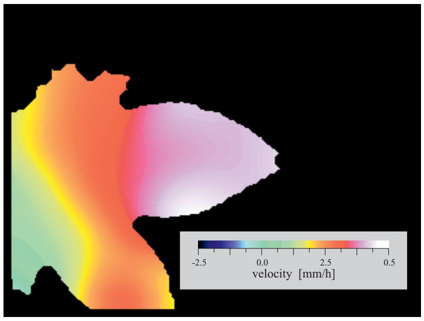
b



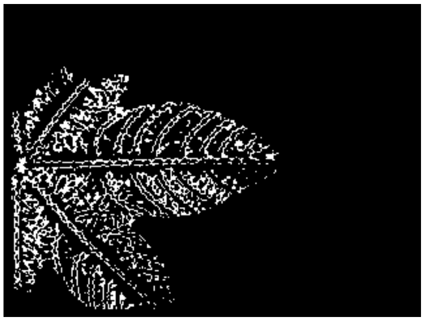
c



d



e



f

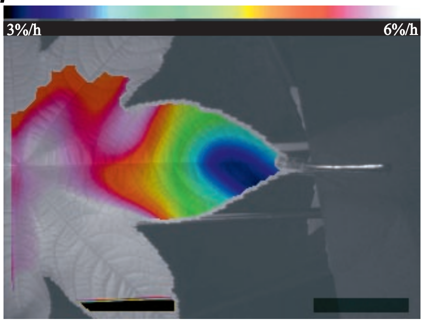


Plate 7: *Six images from the course of a full 2-D growth evaluation:* ***a*** *image from the original time-lapse sequence;* ***b*** *raw velocity field (x-component, color coded) of the leaf surface as calculated from the sequence by a low-level motion estimator (see text);* ***c*** *subset of pixel with a confidence measure above a set threshold;* ***d*** *velocity field when* ***c*** *is considered (masked by the leaf mask);* ***e*** *interpolated velocity field (x-component);* ***f*** *divergence of the velocity field (x- and y-direction); (see also Fig. 33.5, p. 728)*

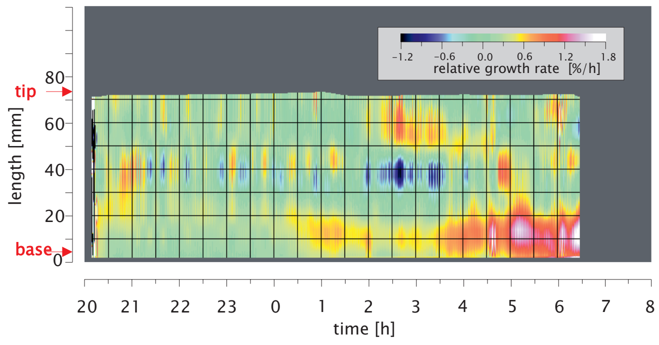


Plate 8: *The growth of the middle part of a Ricinus leaf depicted as a spatiotemporal image; (see also Fig. 33.7, p. 732)*

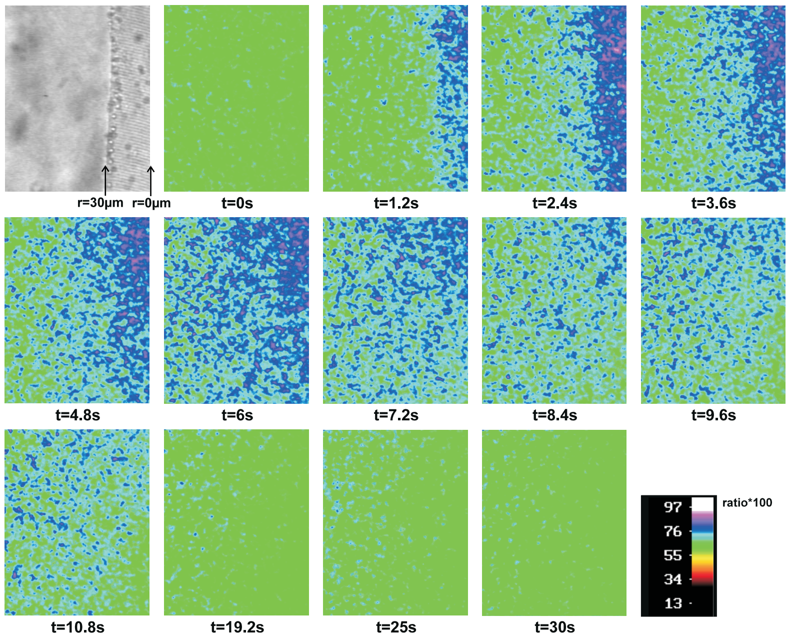


Plate 9: *Time series of a caffeine-induced Ca^{2+}-transient recorded with the Ca^{2+}-sensitive dye Fura-2 (10 µM) using the wavelength pair 340 nm/380 nm; (see also Fig. 34.3, p. 742)*

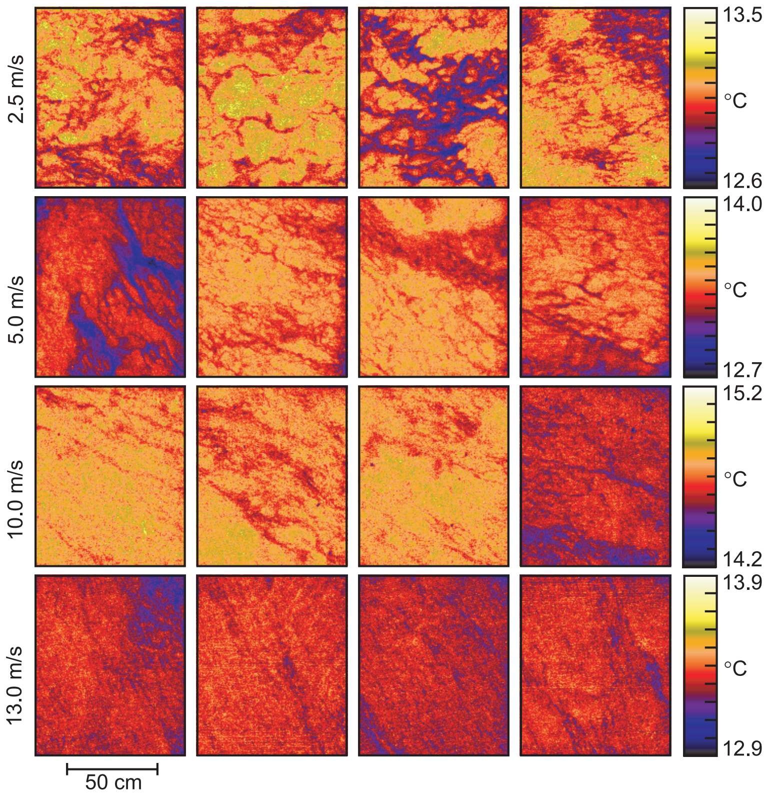


Plate 10: *Infrared images of the water surface at different wind speeds. The image brightness is temperature calibrated. Obviously, the temperature contrast and the size of the structures on the water surface decrease with higher wind speeds; (see also Fig. 35.3, p. 754)*

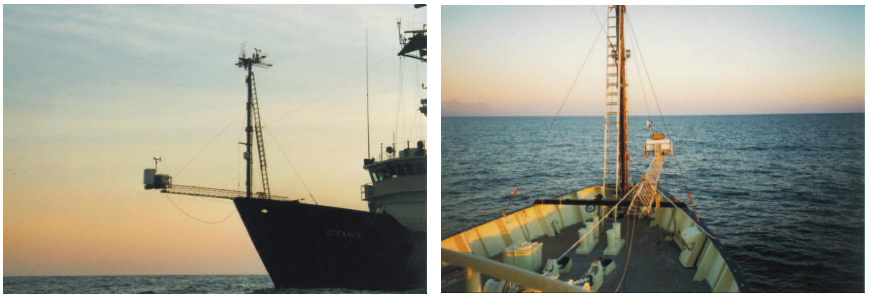


Plate 11: *The CFT instrument is mounted on a 7-m long boom at the bow of the Research Vessel Oceanus during the cruise in the North Atlantic, July 1997; (see also Fig. 35.6, p. 757)*

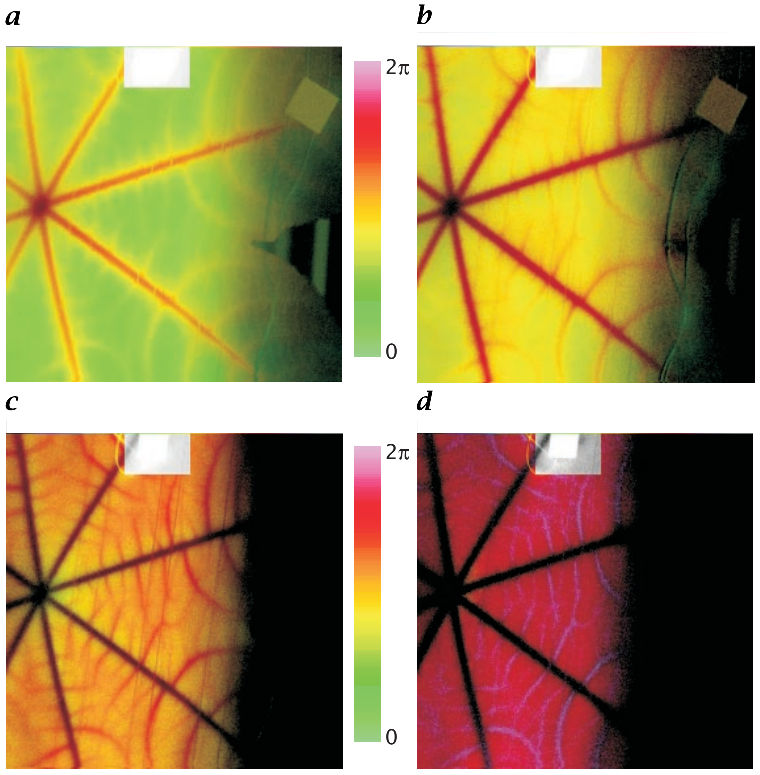
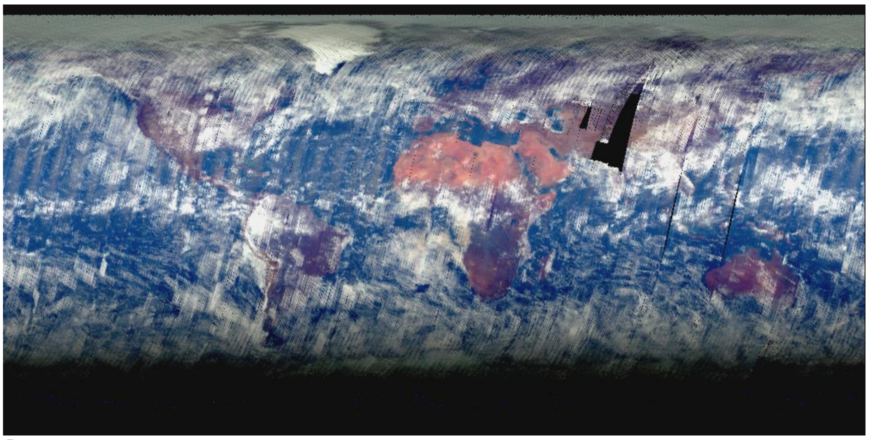


Plate 12: *Parameter images depicting the phase Eq. (36.33) for different input radiation frequencies* ω*:* ***a*** $\omega = 2\pi\, 0.0039$ *rad/s;* ***b*** $\omega = 2\pi\, 0.0078$ *rad/s;* ***c*** $\omega = 2\pi\, 0.0156$ *rad/s;* ***d*** $\omega = 2\pi\, 0.0313$ *rad/s; (see also Fig. 36.6, p. 779)*

a



b

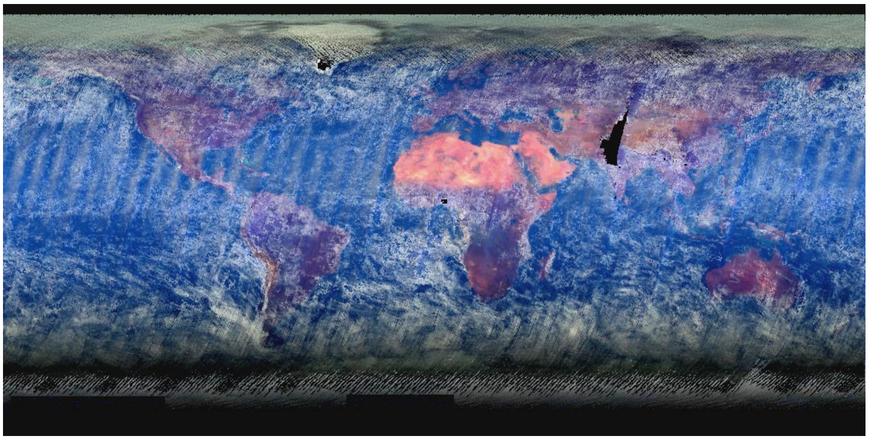


Plate 13: ***a*** *Mean image over a time series of PMD images over 30 days. The clouds can be easily distinguished with their whiteness.* ***b*** *Mean image after cloud removal. Most of the clouds could be located and removed to obtain a cloud free background image; (see also Fig. 37.9, p. 800)*

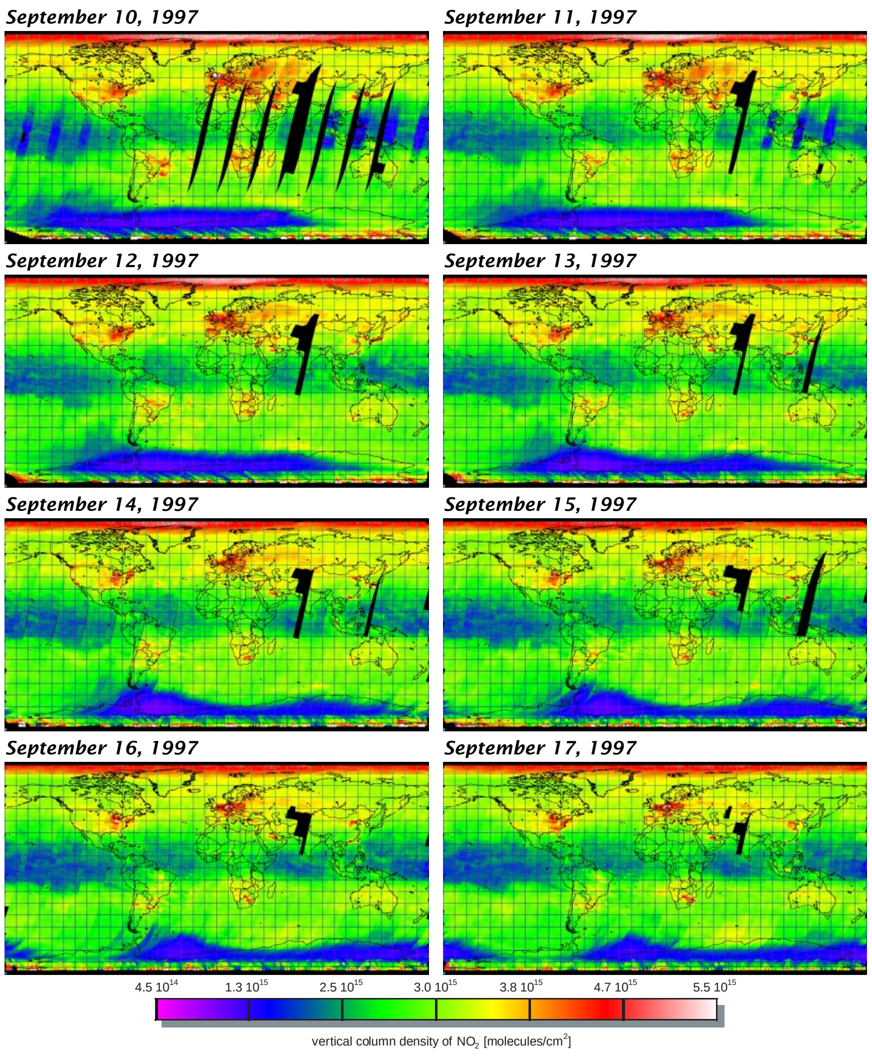


Plate 14: *Time sequence of vertical column densities of NO_2. Large emission plumes can be noticed both over industrialized areas in Europe and North America and over regions where biomass burning occurs (South America, Africa). The time sequence shows the movement of the plumes over a time period; (see also Fig. 37.12, p. 803)*

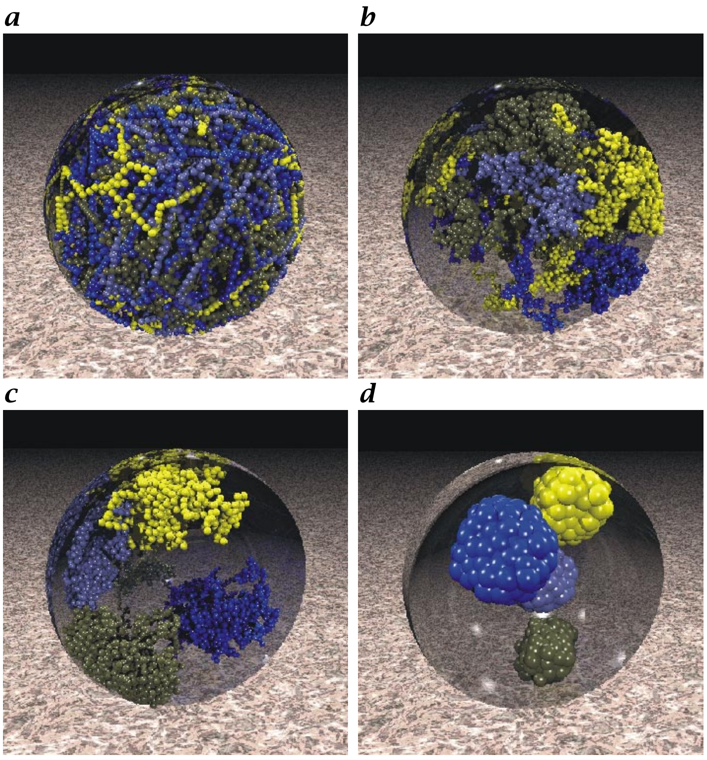


Plate 15: *Computer simulations of all chromosomes in the human interphase cell nucleus. Chromosomes were described by different models, which approximate the chromatin fiber by a polymer chain, folded in different ways. Corresponding to the different condensation levels, a nonterritorial (**a** Vogel and Schroeder-, **b** random-walk bead model) or a territorial (**c** random-walk giant loop-, **d** spherical subdomain model) organization of chromosomes was obtained. For visualization, in each case only four chromosomes are visualized in a spherical nuclear envelope by ray tracing; (see also Fig. 40.1, p. 840)*

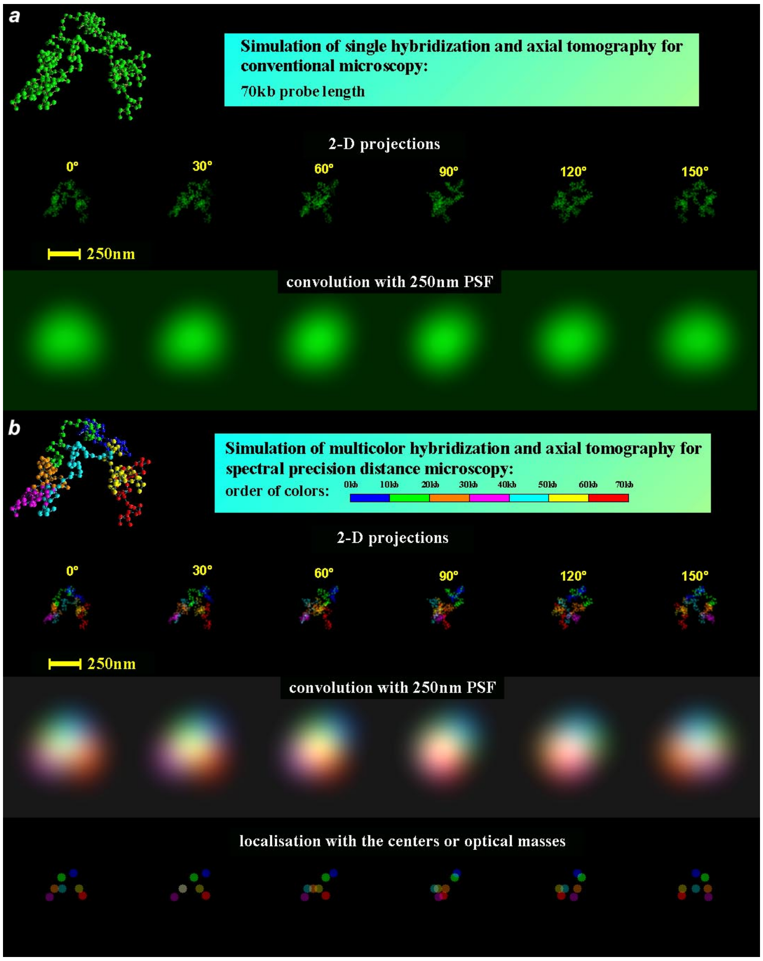

a
Simulation of single hybridization and axial tomography for conventional microscopy:
70kb probe length
2-D projections
0°
30°
60°
90°
120°
150°
250nm
convolution with 250nm PSF
b
Simulation of multicolor hybridization and axial tomography for spectral precision distance microscopy:
order of colors:
0kb
10kb
20kb
30kb
40kb
50kb
60kb
70kb
2-D projections
0°
30°
60°
90°
120°
150°
250nm
convolution with 250nm PSF
localisation with the centers or optical masses

Plate 16: ***a*** *Zig-zag random-walk computer simulation of a 70-kb long nucleosome chain (beads) of chromosomal DNA after single color hybridization. In conventional fluorescence microscopy (corresponding to an assumed convolution with a PSF of 250-nm FWHM), even under various rotation angles no inner structures are detectable;* ***b*** *zig-zag random-walk computer simulation of the same 70-kb long nucleosome chain of chromosomal DNA after hybridization with seven spectrally different signatures (each color segment 10 kbp). Each segment is assumed to be detected by spectrally discriminated imaging. In spectral precision distance microscopy (SPM), the barycenters of the intensities can still be localized after convolution with the PSF of the microscope, so that their distances can be measured. In combination with axial tomography for rotation of the labeled object, the 3-D conformation of the DNA segment can be revealed, although the barycenter distances were considerably below the resolution limit of the microscopy given by the FWHM of the PSF; (see also Fig. 40.4, p. 846)*

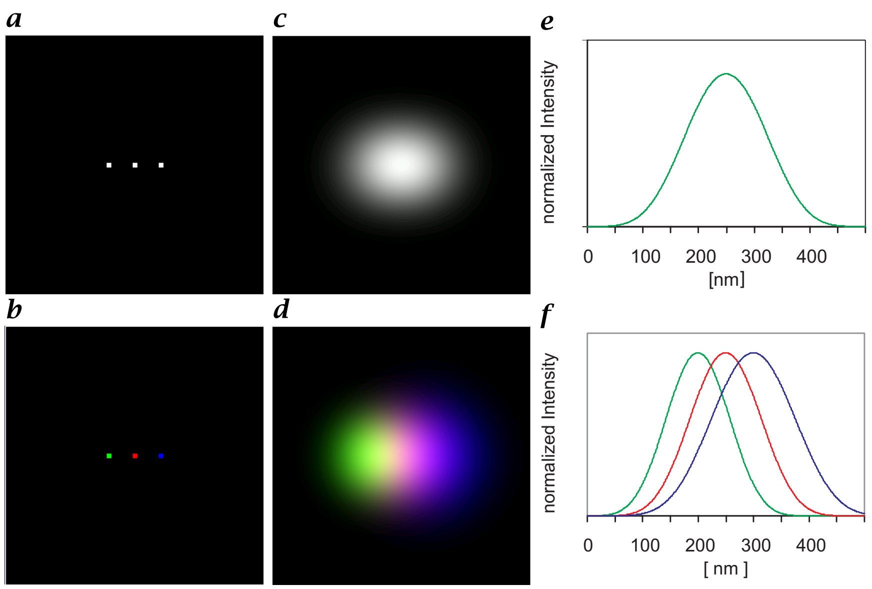


Plate 17: *Example describing the principle of spectral precision distance microscopy (SPM). Three point-like objects are located within 50-nm distance of each other. The three point-like objects are labeled with the same spectral signature in **a** and with three different spectral signatures in **b**. The computed system responses of the objects in **a** and **b** for a confocal microscope with NA = 1.4/63× oil-immersion objective are shown in **c** and **d**. Linescanes through the objects in **c** and **d** are shown in **e** and **f** respectively; (see also Fig. 40.5, p. 847)*

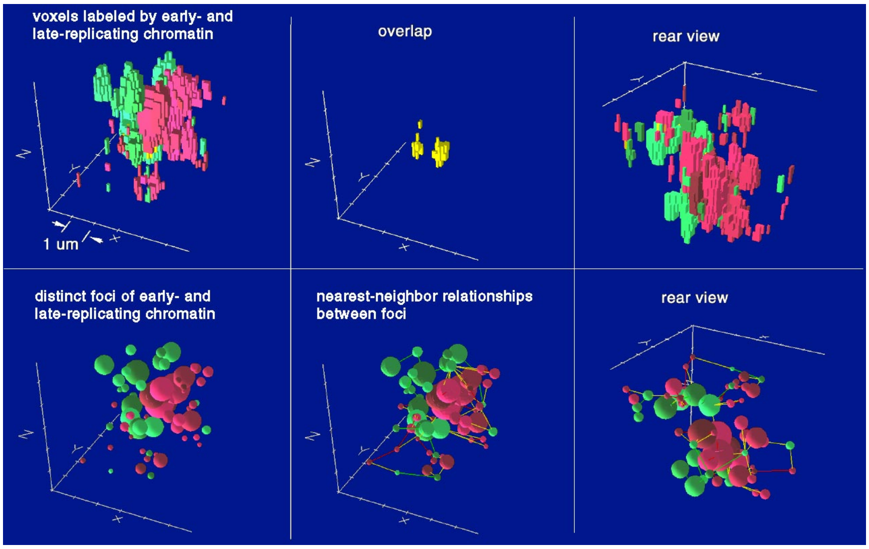

voxels labeled by early- and late-replicating chromatin
overlap
rear view
1 um
distinct foci of early- and late-replicating chromatin
nearest-neighbor relationships between foci
rear view

Plate 18: *Visualization of a segmented chromosome 15 territory in G1 phase. Upper row: the segmented voxels for early-replicating chromatin (dark gray) and late-replicating chromatin (light gray). Lower row: the individual foci found by the segmentation algorithm, shown as spheres with a radius that yields the individual volume found. Distance vectors between nearest neighbors are denoted by lines in the middle and on the right for two different perspectives; (see also Fig. 41.9, p. 875)*