

Roberto Cipolla  
Sebastiano Battiato  
Giovanni Maria Farinella (Eds.)

# Computer Vision

Detection, Recognition and Reconstruction



Springer

Roberto Cipolla, Sebastiano Battiato, and Giovanni Maria Farinella (Eds.)

---

Computer Vision

# Studies in Computational Intelligence, Volume 285

## Editor-in-Chief

Prof. Janusz Kacprzyk  
Systems Research Institute  
Polish Academy of Sciences  
ul. Newelska 6  
01-447 Warsaw  
Poland

E-mail: kacprzyk@ibspan.waw.pl

---

Further volumes of this series can be found on our homepage: [springer.com](http://springer.com)

Vol. 265. Zbigniew W. Ras and Li-Shiang Tsay (Eds.)  
*Advances in Intelligent Information Systems*, 2009  
ISBN 978-3-642-05182-1

Vol. 266. Akitoshi Hanazawa, Tsutomu Miki,  
and Keiichi Horio (Eds.)  
*Brain-Inspired Information Technology*, 2009  
ISBN 978-3-642-04024-5

Vol. 267. Ivan Zelinka, Sergej Celikovský, Hendrik Richter,  
and Guanrong Chen (Eds.)  
*Evolutionary Algorithms and Chaotic Systems*, 2009  
ISBN 978-3-642-10706-1

Vol. 268. Johann M.Ph. Schumann and Yan Liu (Eds.)  
*Applications of Neural Networks in High Assurance Systems*,  
2009  
ISBN 978-3-642-10689-7

Vol. 269. Francisco Fernández de de Vega and  
Erick Cantú-Paz (Eds.)  
*Parallel and Distributed Computational Intelligence*, 2009  
ISBN 978-3-642-10674-3

Vol. 270. Zong Woo Geem  
*Recent Advances In Harmony Search Algorithm*, 2009  
ISBN 978-3-642-04316-1

Vol. 271. Janusz Kacprzyk, Frederick E. Petry, and  
Adnan Yazici (Eds.)  
*Uncertainty Approaches for Spatial Data Modeling and  
Processing*, 2009  
ISBN 978-3-642-10662-0

Vol. 272. Carlos A. Coello Coello, Clarisse Dhaenens, and  
Laetitia Jourdan (Eds.)  
*Advances in Multi-Objective Nature Inspired Computing*,  
2009  
ISBN 978-3-642-11217-1

Vol. 273. Fatos Xhafa, Santi Caballé, Ajith Abraham,  
Thanasis Daradoumis, and Angel Alejandro Juan Perez  
(Eds.)  
*Computational Intelligence for Technology Enhanced  
Learning*, 2010  
ISBN 978-3-642-11223-2

Vol. 274. Zbigniew W. Raś and Alicja Wieczorkowska (Eds.)  
*Advances in Music Information Retrieval*, 2010  
ISBN 978-3-642-11673-5

Vol. 275. Dilip Kumar Pratihar and Lakhmi C. Jain (Eds.)  
*Intelligent Autonomous Systems*, 2010  
ISBN 978-3-642-11675-9

Vol. 276. Jacek Mańdziuk  
*Knowledge-Free and Learning-Based Methods in Intelligent  
Game Playing*, 2010  
ISBN 978-3-642-11679-3

Vol. 277. Filippo Spagnolo and Benedetto Di Paola (Eds.)  
*European and Chinese Cognitive Styles and their Impact on  
Teaching Mathematics*, 2010  
ISBN 978-3-642-11679-7

Vol. 278. Radomir S. Stankovic and Jaakko Astola  
*From Boolean Logic to Switching Circuits and Automata*, 2010  
ISBN 978-3-642-11681-0

Vol. 279. Manolis Wallace, Ioannis E. Anagnostopoulos,  
Phivos Mylonas, and Maria Bielikova (Eds.)  
*Semantics in Adaptive and Personalized Services*, 2010  
ISBN 978-3-642-11683-4

Vol. 280. Chang Wen Chen, Zhu Li, and Shiguo Lian (Eds.)  
*Intelligent Multimedia Communication: Techniques and  
Applications*, 2010  
ISBN 978-3-642-11685-8

Vol. 281. Robert Babuska and Frans C.A. Groen (Eds.)  
*Interactive Collaborative Information Systems*, 2010  
ISBN 978-3-642-11687-2

Vol. 282. Huseyin Taha Sencar, Sergio Velastin,  
Nikolaos Nikolidakis, and Shiguo Lian (Eds.)  
*Intelligent Multimedia Analysis for Security  
Applications*, 2010  
ISBN 978-3-642-11754-1

Vol. 283. Ngoc Thanh Nguyen, Radoslaw Katarzyniak, and  
Shyi-Ming Chen (Eds.)  
*Advances in Intelligent Information and Database Systems*,  
2010  
ISBN 978-3-642-12089-3

Vol. 284. Juan R. González, David Alejandro Pelta,  
Carlos Cruz, Germán Terrazas, and Natalio Krasnogor (Eds.)  
*Nature Inspired Cooperative Strategies for Optimization  
(NICSO 2010)*, 2010  
ISBN 978-3-642-12537-9

Vol. 285. Roberto Cipolla, Sebastiano Battiato, and  
Giovanni Maria Farinella (Eds.)  
*Computer Vision*, 2010  
ISBN 978-3-642-12847-9

Roberto Cipolla, Sebastiano Battiato,  
and Giovanni Maria Farinella (Eds.)

# Computer Vision

Detection, Recognition and Reconstruction



Springer

**Prof. Roberto Cipolla**  
Department of Engineering  
University of Cambridge  
Cambridge, CB2 1PZ  
UK  
E-mail: cipolla@eng.cam.ac.uk

**Dr. Giovanni Maria Farinella**  
Dipartimento di Matematica ed Informatica  
University of Catania  
Viale A. Doria 6, I  
95125 - Catania  
Italy  
E-mail: gfarinella@dmi.unict.it

**Prof. Sebastiano Battiato**  
Dipartimento di Matematica ed Informatica  
University of Catania  
Viale A. Doria 6, I  
95125 – Catania  
Italy  
E-mail: battiato@dmi.unict.it

ISBN 978-3-642-12847-9

e-ISBN 978-3-642-12848-6

DOI 10.1007/978-3-642-12848-6

Studies in Computational Intelligence

ISSN 1860-949X

Library of Congress Control Number: 2010925724

© 2010 Springer-Verlag Berlin Heidelberg

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

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

*Typeset & Cover Design:* Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

[springer.com](http://springer.com)

# **Editors' Biographies**

## **Roberto Cipolla**

Roberto Cipolla obtained the B.A. degree (Engineering) from the University of Cambridge in 1984 and an M.S.E. (Electrical Engineering) from the University of Pennsylvania in 1985. From 1985 to 1988 he studied and worked in Japan at the Osaka University of Foreign Studies (Japanese Language) and Electrotechnical Laboratory. In 1991 he was awarded a D.Phil. (Computer Vision) from the University of Oxford and from 1991-92 was a Toshiba Fellow and engineer at the Toshiba Corporation Research and Development Centre in Kawasaki, Japan. He joined the Department of Engineering, University of Cambridge in 1992 as a Lecturer and a Fellow of Jesus College. He became a Reader in Information Engineering in 1997 and a Professor in 2000. He is Professor of Computer Vision at the Royal Academy of Arts, London (since 2004) and a Director of Toshiba's Cambridge Research Laboratory (since 2007). His research interests are in computer vision and robotics and include the recovery of motion and 3D shape of visible surfaces from image sequences; object detection and recognition; novel man-machine interfaces using hand, face and body gestures; real-time visual tracking for localisation and robot guidance; applications of computer vision in mobile phones, visual inspection and image-retrieval and video search. He has authored 2 books, edited 7 volumes and co-authored more than 300 papers. Professor Cipolla founded (in 2006) and currently directs the International Computer Vision Summer School.

## **Sebastiano Battiato**

Sebastiano Battiato was born in Catania, Italy, in 1972. He received the degree in Computer Science (summa cum laude) in 1995 and his Ph.D. in Computer Science and Applied Mathematics in 1999. From 1999 to 2003 he has leaded the "Imaging" team c/o STMicroelectronics in Catania. Since

2004 he has been a researcher at Department of Mathematics and Computer Science of the University of Catania. His research interests include image enhancement and processing, image coding and camera imaging technology. He published more than 90 papers in international journals, conference proceedings and book chapters. He is a co-inventor of about 15 international patents. He is a reviewer for several international journals and he has been regularly a member of numerous international conference committees. He has participated in many international and national research projects. He is an Associate Editor of the SPIE Journal of Electronic Imaging (Specialty: digital photography and image compression). He is a director (and co-founder) of the International Computer Vision Summer School. He is a Senior Member of the IEEE.

## **Giovanni Maria Farinella**

Giovanni Maria Farinella obtained the degree in Computer Science (egregia cum laude) from University of Catania in 2004. He was awarded a Doctor of Philosophy degree (Computer Vision) in 2008. He joined the Department of Mathematics and Computer Science, University of Catania in 2008 as Contract Researcher. He is Contract Professor of Computer Science, Computer Graphics and Computer Vision at the Academy of Arts, Catania (since 2004) and Associate Member of the Computer Vision and Robotics Research Group at University of Cambridge (since 2006). His research interests lie in the fields of Computer Vision, Computer Graphics, Pattern Recognition and Machine Learning. He has co-authored a number of papers in international journals and international conferences proceedings. He also serves as reviewer international journals and conferences. He founded (in 2006) and currently directs the International Computer Vision Summer School.

# Preface

Computer vision is the science and technology of making machines that see. It is concerned with the theory, design and implementation of algorithms that can automatically process visual data to recognize objects, track and recover their shape and spatial layout.

The International Computer Vision Summer School - ICVSS was established in 2007 to provide both an objective and clear overview and an in-depth analysis of the state-of-the-art research in Computer Vision. The courses are delivered by world renowned experts in the field, from both academia and industry, and cover both theoretical and practical aspects of real Computer Vision problems. The school is organized every year by University of Cambridge (Computer Vision and Robotics Group) and University of Catania (Image Processing Lab). Different topics are covered each year. A summary of the past Computer Vision Summer Schools can be found at: <http://www.dmi.unict.it/icvss>

This edited volume contains a selection of articles covering some of the talks and tutorials held during the first two editions of the school on topics such as Recognition, Registration and Reconstruction. The chapters provide an in-depth overview of these challenging areas with key references to the existing literature.

The book starts with two chapters devoted to introducing the reader to the exciting field of Vision. In Chapter 1 a discussion about the fundamentals of the discipline is presented. Human vision is analyzed and a number of basic principles of biological vision that might be of interest to the machine vision community are identified. Chapter 2 introduces a methodology to evaluate the effectiveness of local features when employed for recognition tasks. A novel mathematical characterisation of the co-variance properties of the features which accounts for deviation from the usual idealised image affine (de)formation model together with a novel metrics to evaluate the features are described.

In Chapter 3 and Chapter 4 computational techniques based on Dynamic Graph Cuts and Discriminative Graphical Models are presented and

employed in problems such as image and video segmentation, pose estimation and context based classification. An overview of the Mutual SubSpace Method and its applications in face and character recognition is presented in Chapter 5. The book continues with three chapters that cover recent approaches for detection, classification and recognition of objects, scenes and activities from images. Specifically, Chapter 6 concentrates on the task of activity recognition by using graphical models which combine information from both object recognition and scene classification. Chapter 7 examines Semantic Texton Forests and evaluates their use for image categorization and semantic segmentation, whereas Chapter 8 focuses on finding a suitable representation that can efficiently capture the intrinsic three-dimensional and multi-view nature of object categories to help the recognition and categorization task.

In Chapter 9 a vision-based system for touch-free interaction with a display at a distance is presented after a deep revision of the state of the art techniques on hand tracking. The problem of tracking multiple objects taking into account multiple views is introduced in Chapter 10.

Finally, two Chapters discussing the problem and existing solutions for 3D reconstruction through multiview and photometric stereo conclude the book.

It is our hope that graduate students, young and senior researchers, and academic/industrial professionals will find the book useful for reviewing current approaches and for teaching Computer Vision, thereby continuing the mission of the International Computer Vision Summer School.

Sicily, December 2009

Roberto Cipolla  
Sebastiano Battiato  
Giovanni Maria Farinella

# Acknowledgements

We would like to take this opportunity to thank all contributors of this book, and all people involved in the organization of ICVSS. A special thanks to Professor Giovanni Gallo for his invaluable encouragement, guidance and support.

# Contents

<b>1</b>	<b>Is Human Vision Any Good? . . . . .</b>	<b>1</b>
	<i>Jan J. Koenderink</i>	
1.1	Introduction . . . . .	1
1.1.1	What Is “Human Vision”? . . . . .	1
1.1.2	How Is Human Vision Implemented? . . . . .	2
1.2	Frameworks . . . . .	5
1.2.1	The Spatial Framework . . . . .	6
1.2.2	The Photometric Framework . . . . .	9
1.3	A Case Study: “Shape from Shading” . . . . .	13
1.3.1	The So Called “Shape from Shading Problem” . . . . .	13
1.3.2	Setting up the Problem . . . . .	15
1.3.3	The Local Shape from Shading Problem . . . . .	18
1.4	Final Remarks . . . . .	24
	References . . . . .	25
<b>2</b>	<b>Knowing a Good Feature When You See It: Ground Truth and Methodology to Evaluate Local Features for Recognition . . . . .</b>	<b>27</b>
	<i>Andrea Vedaldi, Haibin Ling, Stefano Soatto</i>	
2.1	Introduction . . . . .	27
2.2	Empirical Studies of Local Features . . . . .	29
2.2.1	Some Nomenclature . . . . .	30
2.3	Constructing a Rigorous Ground Truth . . . . .	30
2.3.1	Modeling the Detector . . . . .	30
2.3.2	Modeling Correspondences . . . . .	33
2.3.3	Ground-Truth Correspondences . . . . .	34
2.3.4	Comparing Ground-Truth and Real-World Correspondences . . . . .	36
2.3.5	The Data . . . . .	37
2.4	Learning to Compare Invariant Features . . . . .	40

2.4.1	Wide Baseline Motion Statistics . . . . .	41
2.4.2	Learning to Rank Matches . . . . .	41
2.4.3	Learning to Accept Matches . . . . .	43
2.5	Discussion . . . . .	44
	Appendix 1: Calculations . . . . .	44
	Appendix 2: Algorithms . . . . .	45
	References . . . . .	48
<b>3</b>	<b>Dynamic Graph Cuts and Their Applications in Computer Vision . . . . .</b>	<b>51</b>
	<i>Pushmeet Kohli, Philip H.S. Torr</i>	
3.1	Introduction . . . . .	51
3.1.1	Markov and Conditional Random Fields . . . . .	52
3.2	Graph Cuts for Energy Minimization . . . . .	54
3.2.1	The st-Mincut Problem . . . . .	54
3.2.2	Formulating the Max-Flow Problem . . . . .	55
3.2.3	Augmenting Paths, Residual Graphs . . . . .	56
3.2.4	Minimizing Functions Using Graph Cuts . . . . .	56
3.3	Minimizing Dynamic Energy Functions Using Dynamic Graph Cuts . . . . .	58
3.3.1	Dynamic Computation . . . . .	58
3.3.2	Energy and Graph Reparameterization . . . . .	60
3.4	Recycling Computation . . . . .	61
3.4.1	Updating Residual Graphs . . . . .	62
3.4.2	Computational Complexity of Update Operations . . . . .	64
3.5	Improving Performance by Recycling Search Trees . . . . .	65
3.5.1	Reusing Search Trees . . . . .	65
3.5.2	Tree Recycling for Dynamic Graph Cuts . . . . .	66
3.6	Dynamic Image Segmentation . . . . .	67
3.6.1	CRFs for Image Segmentation . . . . .	68
3.6.2	Image Segmentation in Videos . . . . .	70
3.6.3	Experimental Results . . . . .	70
3.6.4	Reusing Flow vs. Reusing Search Trees . . . . .	73
3.7	Simultaneous Segmentation and Pose Estimation of Humans . . . . .	74
3.7.1	Pose Specific CRF for Image Segmentation . . . . .	78
3.7.2	Formulating the Pose Inference Problem . . . . .	82
3.7.3	Experiments . . . . .	85
3.7.4	Shape Priors for Reconstruction . . . . .	87
3.7.5	Discussion . . . . .	91
3.7.6	Summary and Future Work . . . . .	92
3.8	Measuring Uncertainty in Graph Cut Solutions . . . . .	93
3.8.1	Preliminaries . . . . .	94
3.8.2	Computing Min-Marginals Using Graph Cuts . . . . .	97

3.8.3	Min-Marginals and Flow Potentials . . . . .	99
3.8.4	Extension to Multiple Labels . . . . .	100
3.8.5	Minimizing Energy Function Projections Using Dynamic Graph Cuts . . . . .	101
3.8.6	Computational Complexity and Experimental Evaluation . . . . .	101
3.8.7	Applications of Min-Marginals . . . . .	102
	References . . . . .	104
<b>4</b>	<b>Discriminative Graphical Models for Context-Based Classification</b> . . . . .	109
	<i>Sanjiv Kumar</i>	
4.1	Contextual Dependencies in Images . . . . .	109
4.1.1	The Nature of Contextual Interactions . . . . .	110
4.2	Markov Random Field (MRF) . . . . .	112
4.3	Conditional Random Field (CRF) . . . . .	113
4.3.1	Association Potential . . . . .	115
4.3.2	Interaction Potential . . . . .	117
4.4	Parameter Learning and Inference . . . . .	119
4.4.1	Maximum Likelihood Parameter Learning . . . . .	119
4.4.2	Inference . . . . .	121
4.5	Extensions . . . . .	121
4.5.1	Multiclass CRF . . . . .	122
4.5.2	Hierarchical CRF . . . . .	123
4.6	Applications . . . . .	127
4.6.1	Man-Made Structure Detection . . . . .	127
4.6.2	Image Classification and Contextual Object Detection . . . . .	129
4.7	Related Work and Further Readings . . . . .	133
	References . . . . .	133
<b>5</b>	<b>From the Subspace Methods to the Mutual Subspace Method</b> . . . . .	135
	<i>Ken-ichi Maeda</i>	
5.1	Introduction . . . . .	135
5.2	The Subspace Methods . . . . .	139
5.2.1	A Brief History of the Subspace Methods . . . . .	139
5.2.2	Basic Idea . . . . .	139
5.2.3	Subspace Construction . . . . .	140
5.3	The Mutual Subspace Method . . . . .	142
5.3.1	Basic Idea . . . . .	142
5.3.2	Application to Chinese Character Recognition . . . . .	144
5.3.3	Application to Face Recognition . . . . .	147
5.3.4	Application to 3-D Face Recognition . . . . .	148
5.4	Conclusion . . . . .	154
	References . . . . .	155

<b>6</b>	<b>What, Where and Who? Telling the Story of an Image by Activity Classification, Scene Recognition and Object Categorization . . . . .</b>	157
	<i>Li Fei-Fei, Li-Jia Li</i>	
6.1	Introduction and Motivation . . . . .	157
6.2	Overall Approach . . . . .	159
6.3	Literature Review . . . . .	160
6.4	The Integrative Model . . . . .	160
6.4.1	Labeling an Unknown Image . . . . .	163
6.5	Learning the Model . . . . .	164
6.6	System Implementation . . . . .	165
6.7	Experiments and Results . . . . .	166
6.7.1	Dataset . . . . .	166
6.7.2	Experimental Setup . . . . .	166
6.7.3	Results . . . . .	169
6.8	Conclusion . . . . .	169
	References . . . . .	170
<b>7</b>	<b>Semantic Texton Forests . . . . .</b>	173
	<i>Matthew Johnson, Jamie Shotton</i>	
7.1	Introduction . . . . .	173
7.2	Related Work . . . . .	174
7.3	Randomized Decision Forests . . . . .	175
7.3.1	Training the Forest . . . . .	176
7.3.2	Experiments . . . . .	180
7.4	Image Categorization . . . . .	183
7.4.1	Tree Histograms and Pyramid Matching . . . . .	184
7.4.2	Categorization Results . . . . .	186
7.5	Semantic Segmentation . . . . .	188
7.6	Soft Classification of Pixels . . . . .	189
7.7	Image-Level Semantic Constraints . . . . .	191
7.7.1	Categorization Results . . . . .	192
7.7.2	The Image Level Prior . . . . .	192
7.8	Compositional Constraints . . . . .	194
7.9	Experiments . . . . .	194
7.9.1	MSRC21 Dataset . . . . .	196
7.9.2	VOC 2007 Segmentation Dataset . . . . .	200
7.10	Discussion . . . . .	200
	References . . . . .	201
<b>8</b>	<b>Multi-view Object Categorization and Pose Estimation . . . . .</b>	205
	<i>Silvio Savarese, Li Fei-Fei</i>	
8.1	Introduction . . . . .	206
8.2	Literature Review . . . . .	209
8.3	The Part-Based Multi-view Model . . . . .	210

8.3.1	Overview . . . . .	210
8.3.2	Canonical Parts and Linkage Structure . . . . .	211
8.4	Building the Model . . . . .	212
8.4.1	Extract Features . . . . .	213
8.4.2	Form Parts . . . . .	213
8.4.3	Find Canonical Parts Candidates . . . . .	214
8.4.4	Create the Model . . . . .	216
8.5	View Synthesis . . . . .	217
8.5.1	Representing an Unseen View . . . . .	217
8.6	Recognizing Object Class in Unseen Views . . . . .	219
8.6.1	Extract Features and Get Part Candidates . . . . .	220
8.6.2	Recognition Procedure: First Step . . . . .	221
8.6.3	Recognition Procedure: Second Step . . . . .	222
8.7	Experiments and Results . . . . .	222
8.7.1	Experiment I: Comparison with Thomas et al. [63] . . . . .	222
8.7.2	Experiment II: Detection and Pose Estimation Results on the Dataset in [54] . . . . .	223
8.7.3	Experiment III: Detection and Pose Estimation Results on the Dataset in [55] . . . . .	227
8.8	Conclusion . . . . .	227
	References . . . . .	228
<b>9</b>	<b>A Vision-Based Remote Control . . . . .</b>	<b>233</b>
	Björn Stenger, Thomas Woodley, Roberto Cipolla	
9.1	Introduction . . . . .	233
9.2	Prior Work . . . . .	235
9.2.1	Hand Tracking for Human Computer Interfaces . . . . .	235
9.2.2	Commercial Gesture Interface Systems . . . . .	238
9.2.3	Visual Tracking of a Single Object . . . . .	239
9.3	Tracking with Multiple Observers . . . . .	240
9.3.1	Observation Models . . . . .	240
9.3.2	Evaluating Single Observers . . . . .	243
9.3.3	Evaluating Multiple Observers . . . . .	245
9.3.4	Parallel Evaluation . . . . .	246
9.3.5	Cascaded Evaluation . . . . .	247
9.3.6	Dynamic Model Discussion . . . . .	247
9.3.7	Experimental Results . . . . .	248
9.3.8	Individual Observers . . . . .	248
9.3.9	Observer Combinations . . . . .	249
9.3.10	Tracker Evaluation on Selected Combinations . . . . .	252
9.4	Gesture Interface System . . . . .	253
9.4.1	Visual Attention Mechanism . . . . .	253
9.4.2	Tracking Mechanism . . . . .	254
9.4.3	Selection Mechanisms . . . . .	255

9.5	Summary and Conclusion .....	257
	References .....	258
<b>10</b>	<b>Multi-view Multi-object Detection and Tracking .....</b>	<b>263</b>
	Murtaza Taj, Andrea Cavallaro	
10.1	Introduction .....	263
10.2	Problem Formulation .....	265
10.3	Calibration and Fusion .....	266
10.3.1	Single-Level Homography .....	266
10.3.2	Multi-Level Homography .....	268
10.4	Track-First Approaches .....	269
10.4.1	Independent Tracking .....	271
10.4.2	Collaborative Tracking .....	271
10.5	Fuse-First Approaches .....	272
10.5.1	Detection-Based Tracking .....	273
10.5.2	Track-Before-Detect .....	276
10.6	Conclusions .....	278
	References .....	278
<b>11</b>	<b>Shape from Photographs: A Multi-view Stereo Pipeline .....</b>	<b>281</b>
	Carlos Hernández, George Vogiatzis	
11.1	Introduction .....	281
11.2	Multi-view Stereo Pipeline: From Photographs to 3D Models .....	284
11.3	Computing Photo-Consistency from a Set of Calibrated Photographs .....	285
11.3.1	Normalized Cross Correlation for Depth-Map Computation .....	287
11.3.2	Depth Map Estimation .....	288
11.3.3	Photo-Consistency 3D Map from a Set of Depth-Maps .....	292
11.4	Extracting a 3D Surface from a 3D Map of Photo-Consistency .....	292
11.4.1	Multi-view Stereo Using Multi-resolution Graph-Cuts .....	292
11.4.2	Discontinuity Cost from a Set of Depth-Maps .....	294
11.4.3	Graph Structure .....	295
11.4.4	Labeling Cost from a Set of Depth-Maps .....	295
11.4.5	Probabilistic Fusion of Depth Sensors .....	296
11.4.6	Deformable Models .....	299
11.5	Experiments .....	302
11.5.1	Depth Map Evaluation .....	302
11.5.2	Multi-view Stereo Evaluation .....	304

11.5.3	Digitizing Works of Art . . . . .	304
11.6	Discussion . . . . .	307
	References . . . . .	309
<b>12</b>	<b>Practical 3D Reconstruction Based on Photometric Stereo . . . . .</b>	<b>313</b>
	<i>George Vogiatzis, Carlos Hernández</i>	
12.1	Introduction . . . . .	313
12.2	Photometric Stereo with Coloured Light . . . . .	314
12.2.1	Classic Three-Source Photometric Stereo . . . . .	315
12.2.2	Multi-spectral Sources and Sensors . . . . .	316
12.2.3	Calibration . . . . .	317
12.2.4	Comparison with Photometric Stereo . . . . .	317
12.2.5	The Problem of Shadows . . . . .	318
12.2.6	Facial Capture Experiments . . . . .	325
12.2.7	Related Work . . . . .	328
12.3	Multi-view Photometric Stereo . . . . .	330
12.3.1	Related Work . . . . .	331
12.3.2	Algorithm . . . . .	333
12.3.3	Experiments . . . . .	338
	References . . . . .	342
<b>Index . . . . .</b>		<b>347</b>

# List of Contributors

## **Andrea Cavallaro**

School of Electronic Engineering  
and Computer Science  
Queen Mary University of London  
Mile End Road, London E1 4NS, UK  
[andrea.cavallaro@elec.qmul.ac.uk](mailto:andrea.cavallaro@elec.qmul.ac.uk)

## **Roberto Cipolla**

Department of Engineering  
University of Cambridge  
CB2 1PZ, Cambridge UK  
[cipolla@eng.cam.ac.uk](mailto:cipolla@eng.cam.ac.uk)

## **Li Fei-Fei**

Department of Computer Science,  
Stanford University,  
353 Serra Mall, Gates Building,  
CA 94305-9020, Stanford, USA  
[feifeili@cs.stanford.edu](mailto:feifeili@cs.stanford.edu)

## **Carlos Hernández**

Computer Vision Group  
Toshiba Research Cambridge Ltd,  
CB4 0GZ, Cambridge, UK  
[carlos.hernandez@crl.toshiba.co.uk](mailto:carlos.hernandez@crl.toshiba.co.uk)

## **Matthew Johnson**

Nokia Spear Tower, 1 Market Plaza  
CA 94105, San Francisco, USA  
[matthew.3.johnson@nokia.com](mailto:matthew.3.johnson@nokia.com)

## **Jan J. Koenderink**

EEMCS, Delft University of  
Technology  
P.O. Box 5031, 2600GA Delft,  
The Netherlands  
[jan.koenderink@telfort.nl](mailto:jan.koenderink@telfort.nl)

## **Pushmeet Kohli**

Microsoft Research,  
7 J.J. Thomson Ave,  
CB3 0FB, Cambridge, UK  
[pkohli@microsoft.com](mailto:pkohli@microsoft.com)

## **Sanjiv Kumar**

Google Research  
76 Ninth Ave, New York, NY, USA  
[sanjivk@google.com](mailto:sanjivk@google.com)

## **Li-Jia Li**

Department of Computer Science  
Stanford University  
353 Serra Mall, Gates Building  
CA 94305-9020, Stanford, USA  
[lijiali@cs.stanford.edu](mailto:lijiali@cs.stanford.edu)

## **Haibin Ling**

Center for Information Science &  
Technology  
Department of Computer &  
Information Sciences

324 Wachman Hall,  
Temple University  
1805 N. Broad St.,  
Philadelphia, PA 19122  
[hbling@temple.edu](mailto:hbling@temple.edu)

**Ken-ichi Maeda**  
Corporate Research &  
Development Center  
Toshiba Corporation  
1 Komukai-Toshiba-Cho, Kawasaki,  
212-8582, Japan  
[ken.maeda@toshiba.co.jp](mailto:ken.maeda@toshiba.co.jp)

**Silvio Savarese**  
Electrical and Computer Engineering  
University of Michigan  
1301 Beal Avenue, Ann Arbor,  
MI 48109-2122, USA  
[silvio@umich.edu](mailto:silvio@umich.edu)

**Jamie Shotton**  
Microsoft Research,  
7 J.J. Thomson Ave,  
CB3 0FB, Cambridge, UK  
[jamie@shotton.org](mailto:jamie@shotton.org)

**Stefano Soatto**  
Department of Computer Science  
University of California  
at Los Angeles  
Boelter Hall 3531D, 405 Hilgard Ave  
Los Angeles, CA 90095-1596, USA  
[soatto@ucla.edu](mailto:soatto@ucla.edu)

**Björn Stenger**  
Computer Vision Group  
Toshiba Research Cambridge Ltd,  
CB4 0GZ, Cambridge, UK  
[bjorn@cantab.net](mailto:bjorn@cantab.net)

**Murtaza Taj**  
School of Electronic Engineering and  
Computer Science  
Queen Mary University of London  
Mile End Road, London E1 4NS, UK  
[murtaza@elec.qmul.ac.uk](mailto:murtaza@elec.qmul.ac.uk)

**Philip H.S. Torr**  
Department of Computing  
Oxford Brookes University  
Wheatley, Oxford, OX33 1HX, UK  
[philiptorr@brookes.ac.uk](mailto:philiptorr@brookes.ac.uk)

**Andrea Vedaldi**  
Department of Computer Science  
University of California  
at Los Angeles  
Boelter Hall 3531D, 405 Hilgard Ave  
Los Angeles, CA 90095-1596, USA  
[vedaldi@cs.ucla.edu](mailto:vedaldi@cs.ucla.edu)

**George Vogiatzis**  
Aston University  
Birmingham, UK  
[g.vogiatzis@aston.ac.uk](mailto:g.vogiatzis@aston.ac.uk)

**Thomas Woodley**  
Department of Engineering  
University of Cambridge  
CB2 1PZ, Cambridge UK  
[tew32@cam.ac.uk](mailto:tew32@cam.ac.uk)

# Chapter 1

## Is Human Vision Any Good?

Jan J. Koenderink

**Abstract.** Human vision is often referred to as an “existence proof” for challenging targets of machine vision. But in some areas machine vision evidently “beats” human vision, so the questions arise: Is human vision any good, will it be supplanted by machine vision for most tasks soon? I analyze human vision with the aim to provide an answer to such questions. Does machine vision still have anything to learn from human vision? I identify a number of basic principles of biological vision that are likely to be of interest to the machine vision community.

### 1.1 Introduction

On May 11<sup>th</sup>, 1997 at 3:00PM EDT, game #6 of the match of Garry Kasparov, the greatest (human!) player in the history of chess, against IBM’s Deep Blue Supercomputer put the final score at [kasparov 2.5 : deep blue 3.5]. So much for chess [1]. Are human chess players any good? Is vision like chess in this respect?

In order to answer the question one needs to make more precise what is exactly being meant by “human vision” and identify potentially interesting aspects. In an attempt to learn from human vision one needs an overview of the various hardware and algorithmic structures that implement human vision. One might (naïvely) believe that it would suffice to ask physiologists and psychologists for the required information. Such is not the case. The best that can be done is to offer creative guesses on the basis of a panoramic knowledge of these fields, framed in the language understood by engineers. This is my aim in this chapter.

#### 1.1.1 What Is “Human Vision”?

“Human vision” (or, more general, “biological vision”) stands for an extensive bundle of human or animal capabilities of mutually very different types. The simplest of

---

Jan J. Koenderink  
EEMCS, Delft University of Technology, The Netherlands  
e-mail: [jan.koenderink@telfort.nl](mailto:jan.koenderink@telfort.nl)

these are hardly worth the name “vision”, these include many optoreflexes found in the simplest animals, but also present in man. Though vital for survival, they hardly involve anything beyond wiring up a photocell to some effector. More complicated cases of “*optically guided behavior*” include orientation/navigation on the basis of optic flow and so forth. Such mechanisms make up a major part of the “visual system”. This part can be understood in the engineering sense and are already emulated in robotic systems. Here machines are going to beat man, the waiting is for the first soccer match of a human team to a robot team that will be lost by the men. In this chapter I am not so much interested in this aspect of vision.

Most of “optically guided behavior” goes on below the level of awareness. This makes sense because of the real-time aspect and the pre-cognitive nature of overt bodily actions. Just try to walk through consciously figuring out the right signals to be sent to your muscular-skeletal system on the basis of the pattern of photons impinging on your retinas and you will understand why. Here I will narrow down the definition of “vision” to the level of awareness, that is *optically induced experience*. Such experiences do not primarily influence your actual (real-time) behavior as in optically guided behavior, but they are relevant for your *potential* behavior. Another way to put this is to say that visual experiences lead to knowledge. For instance, on leaving the door you may grab an umbrella or a raincoat because an earlier (maybe much earlier) look out of the window taught you that it “looked like rain”. Or you might do neither and remark “I don’t mind to get wet” to a companion. Thus there is *no direct link with action* as there is in optically guided behavior.

As you open your eyes the world appears to you, you cannot voluntarily choose not to see. Visual experiences are *presentations* that *happen to you*, much like sneezing. Through your presentations (including those of your other modalities) you are aware of your world. You are visually aware of the world itself, rather than your thoughts of the world. That is why you are responsible for your thoughts but not for your presentations. Presentations are pre-cognitive. They are “world-directed” though, the technical term is “intentional”.

### **1.1.2 How Is Human Vision Implemented?**

I will mainly address the level of the brain (including the retinas) here, although the eye with its optics (cornea, lens, waveguides with photopigments), the eyeball with its muscles, as well as the musculature of head and body play major roles in everyday vision. One learns about the visual system from observations of generic humans (experimental psychology, psychophysics), patients (neuropsychiatry), and dead bodies (neuroanatomy). Nowadays one also records crude brain activity in healthy volunteers and (sporadic) epileptic patients during operations. Most of the more spatiotemporally precise observations are from electrophysiological experiments on animals, sometimes (technically) asleep, sometimes in a waking or (very rarely) even behaving state. Although huge amounts of observations are available, the brain is such a complicated structure that it is fair to say that a synthesis of all this knowledge is sadly lacking. One has to make do with lacunary data, usually

available in potentially misleading formats. What I attempt to do here is to identify some basic principles.

**The global neural structure.** The global structure of the visual system is made up of a large number of mutually highly (two way) interconnected areas, roughly branching out from the input stage as a hierarchical tree. As one moves into the hierarchy the resolution decreases, in the areas with considerable resolution one finds a retinotopic ordering, probably reflecting a tendency towards wiring economy. The interconnection of the areas is so high that one can do little more than identify a (very) rough division of labor. The two way wiring suggests that the processing is certainly not limited to a “bottom up” stage.

*The principle of locality.* Most of the wiring inside an area is short range. Most of the processing appears to be local, though with important “top down” (thus probably somewhat global) modulation.

*The principle of local selectivity.* Most of the short range wiring inside an area is highly selective. A common structure is “center–surround”, essentially an isotropic Laplacean operator. Directional wirings are also common, essentially directional derivative operators in space or space–time. Directional units are typically connected to units of similar specificity in their environment.

*The principle of global selectivity.* Long range (area–to–area) wiring is (at least approximately) somatotopic, that is selective in the sense that topological structure is conserved.

**Speculative interpretation.** It is hard not to regard the various anatomically and electrophysiologically distinct areas as functional subunits, dedicated to one or more subtasks. The areas early in the stream (closest to the input) are best known and are likely to be dedicated to various “image processing” tasks. The units are likely to function largely independent of each other, possibly at quite different parameter settings (due to local “adaptation”). The “glue” may be provided by the top down queries and (implicitly) through the somatotopic connection to other areas.

A likely interpretation is to regard an area as a “geometry engine” [2], implementing a basis for differential geometric queries. The units would compute a “jet space” composed of directional derivatives up to a certain order (about 4 seems right), possibly at various scales. Then local algorithms might compute various differential invariants (algebraic combinations of derivatives, e.g., curvature) and local statistical measures (e.g., structure tensors) that would be meaningful entities for top down queries. The specific local wiring would be needed to implement geometrical entities like covariant derivatives, curvature tensors, and so forth. This means that a “signal” (e.g., a curvature tensor) cannot be carried by a single nerve fiber, it has to be carried by a (small) ensemble of these. This again entails that present electrophysiological methods all fail to address the meaningful structure of the areas.

The local jets (essentially truncated Taylor series describing local structure) are simply data structures “sitting there”, waiting to be queried, they are not necessarily sent upstream. Think of this structure as “stuck” in the area as a sample of the world

“in brain readable form”. It is available, like a footprint in the sand of a beach. I see no essential difference here even though the latter is “outside” and the former “inside the head”. The queries can be of various nature and may expect the jets to be available as local parameters to be used in computing an answer. What is sent upstream by the area is a mere summary account of its overall state. Thus the “bottom up” stream narrows down to what a top level might use to check the present “gist”. Details are guessed, perhaps to be checked through a (focussed) stream of “top down” queries.

**The global functional structure.** The functional structure of the visual system (here “visual system” is meant in a functional sense distinct from the level of the neural substrate) can only be addressed via observations of the interaction of the agent with its world. Thus one needs the observations of experimental psychology or psychophysics. A few observations stand out as especially basic.

*The intentionality of visual experiences.* Presentations are world-directed, with the immediate implication that they cannot be computed in a purely bottom up fashion. They must be imposed by the agent, hence derive from a centrifugal (“top down” suggests a brain implementation) or “probing” process. Perception is *active*, it is an outward directed probing rather than the passive reception of “data”. Without probing there would be no “data”, but only structure (see below).

*The rock bottom of Gestalts.* The smallest parts of presentations (though not necessarily in a spatial sense) are the Gestalts. There is no way to look “into” a Gestalt. The Gestalts are to the human observer what “releasers” are to animals. They are where the buck stops in “explaining” the available structure, avoiding an infinite regress. They both limit and enable presentations.

**Speculative interpretation.** Much of presentation is essentially “controlled hallucination”. Although the word “hallucination” has a bad ring to it, “hallucination” has the advantage to resolve the problem of world-directedness in one go. Intentionality is there from the start, no need to compute it—which is an impossibility anyway. Hallucinations can be “controlled” through comparison with the *facts*, which are *records of optical structure*. Thus the system needs top down queries in order to keep its hallucinations in tune with its world. This *modus operandus* has several advantages. The “binding problem” vanishes since nothing like “binding” is needed; lacking data are only a lack of constraint on the current hallucination; likewise inherent ambiguity (typical for “Shape From X” algorithms) simply means a partial constraint. The presentations are always complete and unique by construction (for hallucinations are constructions). The only problem is how to “hallucinate” productively. Usually this is no problem since most moments are much like the previous ones. Here the generic knowledge of the agent world (“frames”, “scripts”, “background”, “Bayesian priors”, etc.) comes into play. It enables the “gist” that seeps in from the bottom up to be used effectively. A wrong hallucination is not a disaster, the agent is almost certain to win any “twenty questions game” with its world. Think of wrong hallucinations as vehicles for learning, much like the hypotheses of scientific research (see below). In that sense the controlled hallucination implements a

selective *probing* process. When probing meets resistance due to unexpected mismatches the agent gains information from direct contact with the world in the context of the (intentional) probing. *No gain without pain*: the mismatches are required in order to be able to gain information at all.

*The “Sherlock Model”.* An apt model for the process of “controlled hallucination” is that so expertly wielded by the famous Sherlock Holmes [3]. The “solution” to a case cannot be “computed” for even the input is undefined. Anything (e.g., a discarded cigarette butt, a hair in the soup, an odd noise at noon) can be either a valuable clue or totally irrelevant. There is no end to this as the world is infinitely structured. It is only a plot that enables Holmes to interpret random structures as clues, or even to actually look for them (is there a bloodstain on the curtain, did someone cut a rose, etc.). Moreover, the plot enables such clues to be interpreted, i.e., to become data (meaningful) instead of structure (meaningless). Questions are like computer formats in that they define the meaning of possible answers. The plot is not “computed from data” either. It is freely invented by Holmes, on the basis of his prior experience and his assessment of the “gist” of the scene. If it doesn’t work Holmes discards it for another. How many possibilities are there anyway (the butler did it, or maybe the countess, etc.)? Holmes’ method is not different from the way the sciences operate either. No theory was ever “computed from data”! They are freely invented by people like Einstein and checked against the observations. They lead to novel observations to be made (“probing nature”) as further checks on the theory. Theories that don’t check out are discarded for other (equally freely invented) ones. Thus theories, plots and presentations are subject to a thoroughly Darwinian selection process that soon weeds out ones that fail to explain the world. The ones that remain are current best bets of what the world is like, or—perhaps more apt, the currently most effective *user interfaces* of the agent (scientist, criminal investigator, visual observer, ...) to their world. To the user, the interface is what the world is like, what they *understand* (having constructed it themselves). For the user there is nothing understandable beyond the interface. In that sense perceptions are “mental paint”.

The very idea of “bottom up processing” makes no sense in criminal investigation, nor in science. It makes no sense in visual perception either, despite the fact that so many professionals (from philosophy, physiology, psychology and machine vision alike) remain true believers. This (abortive) model assumes that the world is a well defined place even in the absence of any observer and that it is the task for visual systems to compute a representation of it. Thus one can conveniently assess visual systems, their representations should replicate the world in as detailed a manner as possible. Such a weird notion applies (at best) to zombies (no intentionality) in limited settings.

## 1.2 Frameworks

In discussing vision one has to break down the problems in more or less coherent chunks. In this chapter I discuss two generic “frameworks”, but I am by no means

complete, nor did I make a serious attempt at a principled breakdown of the whole of vision. I don't doubt that such an endeavor is both possible and profitable though. I pick these two examples because they lead to a similar formal structure ("geometry"), which enables me to keep the formalism at bay.

### 1.2.1 The Spatial Framework

"Space", in the senses of "configuration", "shape" or "possibility of movement", is one of the very backbones of presentations. There are many directions from which one might approach the topic of "space", various of them of a very fundamental nature. Here I simply consider one possibility (arguably the simplest case), the structure of visual space as related to the physical space surrounding a single vantage point ("cyclopean, stationary observer"). I will consider the topology of the visual field to be "given" (so called problem of "local sign" [4]). These are strong assumptions, so we're dealing with a toy system. (However, I know of no machine vision work that seriously deals with the local sign problem.)

If you turn the whole world about the vantage point, the observer can undo this change through a voluntary eye movement, a rotation of the eye about its center. In the absence of additional data (e.g., a gravity sensor) such changes generate no information. The global optical structure shifts but the (all important) local spatial configurations are not affected.

If you scale the whole world about the vantage point the optical structure remains invariant. In the absence of additional data (e.g., the presence of Gulliver—who doesn't scale—in Lilliput and Brobdignac) such changes cannot be recorded at all.

We conclude that the optical structure available to a stationary, cyclopean observer is invariant against the group of rotation–dilations about the vantage point. It is easy enough to construct geometries that implement such invariance. Here is an example, consider the Riemann space [5] with line element (metric)

$$ds^2 = \frac{dx^2 + dy^2 + dz^2}{x^2 + y^2 + z^2}. \quad (1.1)$$

This metric is evidently invariant against arbitrary rotation–dilations about the origin  $\{x, y, z\} = \{0, 0, 0\}$ . Transforming to polar coordinates

$$\rho = \sqrt{x^2 + y^2 + z^2}, \quad (1.2)$$

$$\vartheta = \arccos(z), \quad (1.3)$$

$$\varphi = \arctan(x, y), \quad (1.4)$$

and setting

$$\zeta = \log \frac{\rho}{\rho_0}, \quad (1.5)$$

where  $\rho_0$  is an arbitrary unit of length ("yardstick") you obtain

$$ds^2 = d\zeta^2 + d\vartheta^2 + \sin^2 \vartheta d\varphi^2 = d\zeta^2 + d\Omega^2, \quad (1.6)$$

where  $d\Omega^2$  is the line element (metric) of the unit sphere  $\mathbb{S}^2$ . Since the yardstick is arbitrary, the depth coordinate  $\zeta$  indicates a point on the affine line  $\mathbb{A}$  and we conclude that this space is a vector bundle  $\mathbb{S}^2 \times \mathbb{A}$  with base space  $\mathbb{S}^2$  and fibers  $\mathbb{A}$ .

Notice that points  $\{\zeta_1, \vartheta, \varphi\}$  and  $\{\zeta_2, \vartheta, \varphi\}$  are on the same “visual ray” and thus are imaged on the same “pixel”. We conclude that such points are *coincident* in the (physical) *visual field*, though they may be distinct in the (mental) *visual space* of the observer. An example is a glyph like “ $\times$ ” where I may “see” the upslope “ $/$ ” as being in front of the downslope “ $\backslash$ ”, thus the point “.” where they intersect as *two* points, one in front of the other, perhaps symbolized as “ $\odot$ ”. Of course I might as well “see” the upslope “ $/$ ” as being *behind* the downslope “ $\backslash$ ”, the depth order being fully idiosyncratic. “Visual space” is a mental entity where the mind may shift the depths  $\zeta_{1,2,\dots}$  on the visual rays (directions, points of  $\mathbb{S}^2$ ) as if they were beads on strings.

In order to deal with this in a formal manner you may redefine the metric in such a way that points like  $\{\zeta_1, \vartheta, \varphi\}$  and  $\{\zeta_2, \vartheta, \varphi\}$  that are on the same “visual ray” are assigned *zero distance* whereas still considered *different*. In the tradition of geometry such points would have to be designated “parallel”. This is fully analog to the usage in the case of planes in space. Generically two planes subtend a finite angle (their distance in the angle metric), but it may happen that this angle vanishes without the planes being coincident. In that case one designates the planes to be “parallel”.

The way to bring this about is to make the depth dimension *isotropic* [6]. On the “isotropic line” any two points subtend mutual distance zero. This is often useful in science, perhaps the best known example being the special theory of relativity where the light cones have isotropic generators. In a convenient formalism I introduce the “dual imaginary unit  $\varepsilon$ ”, where  $\varepsilon$  is defined as the nontrivial solution of the quadratic equation  $x^2 = 0$ . That is to say, you have  $\varepsilon^2 = 0$ ,  $\varepsilon \neq 0$ . The numbers  $u + \varepsilon v$  where  $u, v \in \mathbb{R}$  are known as the “dual (imaginary) numbers”, an extension of the real number line, much like the conventional imaginary numbers with imaginary unit  $i$ , where  $i^2 = -1$ . One easily proves that neither  $\varepsilon > 0$  nor  $\varepsilon < 0$  whereas  $\varepsilon \neq 0$ . Thus the “Law of the Excluded Third” does not work for the dual number system and one has to adopt intuitionistic logic, which is probably as well in an engineering context. Engineering “proofs” are by nature *constructive*, “proofs by contradiction” play no role.

Writing the metric as

$$ds^2 = d\Omega^2 + \varepsilon^2 d\zeta^2, \quad (1.7)$$

solves our problem. Points on different visual rays have finite distances (simply their angular separation), whereas points on a single ray have mutual distance zero, even if they are distinct. However, in the latter case one might define them to subtend the “special distance”  $\zeta_2 - \zeta_1$ . This is indeed a useful distance measure because it is invariant against arbitrary rotation–dilations. Notice that the special distance applies only to points with zero regular distance. Then one may define “the” distance as either the regular distance or (in case the regular distance vanishes) the special distance.

### 1.2.1.1 The Case of Narrow Visual Fields

In the case of narrow visual fields the formalism can be simplified. Let the main line of sight be in the Z-direction ( $\vartheta \ll 1$ ). Then the angular coordinates  $\{\vartheta, \varphi\}$  may be replaced with the “Riemann normal coordinates”  $\{\xi, \eta\}$  [5]:

$$\xi = \vartheta \cos \varphi, \quad (1.8)$$

$$\eta = \vartheta \sin \varphi. \quad (1.9)$$

We obtain a space  $\mathbb{E}^2 \times \mathbb{J}$  (where  $\mathbb{E}^2$  denotes the Euclidian plane and  $\mathbb{J}$  the isotropic affine line). The 8-parameter group

$$\xi' = a(+\xi \cos \beta + \eta \sin \beta) + c_\xi, \quad (1.10)$$

$$\eta' = a(-\xi \sin \beta + \eta \cos \beta) + c_\eta, \quad (1.11)$$

$$\zeta' = f_\xi \xi + f_\eta \eta + g \zeta + h, \quad (1.12)$$

is not unlike that of the similarities (thus including congruencies and movements) of Euclidian space  $\mathbb{E}^3$ , except for the fact that the latter group is only a 7-parameter group. The group scales distances by the amount  $a$ , which may be regarded as due to dilations, because it implements pseudo-perspective scaling; the parameter  $\beta$  results from rotations about the viewing direction, the parameters  $\{c_\xi, c_\eta\}$  from rotations about axes orthogonal to the viewing direction; the parameter  $h$  from a depth shift. The parameters  $\{f_\xi, f_\eta\}$  and  $g$  have been added because they conserve distance and special distance, for if  $(\xi_2 - \xi_1)^2 + (\eta_2 - \eta_1)^2 > 0$  you have (due to  $\varepsilon^2 = 0$ )

$$((\xi'_2 - \xi'_1)^2 + (\eta'_2 - \eta'_1)^2) = a^2 ((\xi_2 - \xi_1)^2 + (\eta_2 - \eta_1)^2), \quad (1.13)$$

whereas for  $(\xi_2 - \xi_1)^2 + (\eta_2 - \eta_1)^2 = 0$  you have

$$(\zeta'_2 - \zeta'_1) = g(\zeta_2 - \zeta_1). \quad (1.14)$$

This group of similarities defines (in the sense of Felix Klein) the Cayley–Klein space (one of 27) with a single isotropic dimension. The space has a parabolic distance measure (like Euclidian space), but unlike Euclidian space also a parabolic angle measure. This accounts for the additional group parameter: one may scale either distances or angles (or both), whereas Euclidian angles cannot be scaled because periodic (elliptic angle measure).

Since the similarities in the “image plane” (the  $\{\xi, \eta\}$ -plane) are trivial, as are depth shifts, the subgroup

$$\xi' = \xi, \quad (1.15)$$

$$\eta' = \eta, \quad (1.16)$$

$$\zeta' = f_\xi \xi + f_\eta \eta + g \zeta, \quad (1.17)$$

is perhaps of most immediate interest. It is the group of “bas-relief ambiguities” identified for the “Shape From Shading” problem of machine vision. Apparently the shading setting is irrelevant here, this transformation follows from a very general analysis of stationary, cyclopean vision. The parameter  $g$  immediately scales the depth of relief, whereas the parameters  $\{f_\xi, f_\eta\}$  describe “additive planes”, formally they describe isotropic rotations.

Consider a rotation

$$\xi' = \xi, \quad (1.18)$$

$$\zeta' = f\xi + \zeta, \quad (1.19)$$

in a constant  $\eta$  plane. The “frontoparallel” line  $\zeta = 0$  is transformed into the slanted line  $\zeta = f\xi$ , which has the isotropic slope angle  $f$ . (This is a good angle measure because the transformation changes the slope of arbitrarily slanted lines by the same amount.) Apparently the slope angle varies between  $\pm\infty$  and is not periodic. Thus you can’t rotate the line “upside down”. There is no “turning around” in visual space! Notice that this is exactly what is required because these rotations are in mental space: if you see the front of an object you can’t see its back, no matter how you shift the depth “beads” along their visual ray “strings”. The formalism perfectly captures the condition of a stationary, cyclopean observer.

The geometry and differential geometry of this “visual space” has been developed in detail during the first half of the 20<sup>th</sup>c. (Not in the context of vision, but as a purely formal endeavor [7].) The resulting geometry is as rich as that of the familiar Euclidean space  $\mathbb{E}^3$ , though with some surprises. Calculations are typically much simpler than they are for analogous problems in Euclidean space, the main reason being algebraic. The full Taylor expansion of a function  $F(u + \varepsilon v)$  being

$$F(u + \varepsilon v) = F(x) + \varepsilon F'(x)v, \quad (1.20)$$

(no higher order terms!) really makes life easy. Especially, the trigonometric functions become

$$\sin \varepsilon x = \varepsilon x, \quad (1.21)$$

$$\cos \varepsilon x = 1, \quad (1.22)$$

which enormously simplifies numerous calculations in geometry.

### 1.2.2 The Photometric Framework

In the “photometric framework” one deals with an “image plane”, which is a Euclidian plane  $\mathbb{E}^2$ , and a scalar “intensity” field  $I(x, y)$  (say). The intensity could be the irradiance of the image plane due to some optical system for instance.

**The intensity domain.** I will only consider two generic properties of the “intensity”:

- the intensity is positive  $I \in \mathbb{R}^+$  (I consider zero values singular);
- the intensity is a photometric quantity, i.e., its physical dimension is not length but involves radiant power in some way.

In the context of machine vision one typically doesn't care much about physical dimensions, thus it is convenient to decide on some standard intensity  $I_0$  say, and redefine intensity as the dimensionless quantity  $\bar{I} = I/I_0$ .

In the absence of any prior knowledge the Bayesian prior for the intensity is hyperbolic:

$$P(\bar{I}) d\bar{I} = \frac{d\bar{I}}{\bar{I}}, \quad (1.23)$$

thus the *natural* representation of intensity is logarithmic, for then the prior probability density is a uniform density. Consequently I redefine intensity yet another time:

$$J = \log \bar{I} = \log \frac{I}{I_0}. \quad (1.24)$$

Since the fiducial intensity  $I_0$  is arbitrary, the  $J$ -domain fails a natural origin. One concludes that the natural representation of the intensity domain is the affine line  $\mathbb{A}$ .

Apparently the objects that one deals with in the photometric framework are cross sections of the fiberbundle  $\mathbb{E}^2 \times \mathbb{A}$  with the image plane as base space and the intensity domain as fibers. Such cross sections are conveniently referred to as “images”.

**The topological structure.** In most cases the image will not be defined over the whole of the image plane, though the actual size of the available image is often irrelevant. For most purposes one may define a “region of interest”, such that anything outside the region of interest does not affect the calculation. I will refer to the size of the region of interest as the “outer scale” of an image.

In work of a theoretical nature one often thinks of the intensity as defined on any point and one writes  $J(\mathbf{r})$ , with  $\mathbf{r} \in \mathbb{E}^2$ , whereas in work of a practical nature one considers intensities “pixel values” and writes  $J_{ij}$ , with  $ij \in \mathbb{Z}^2$ . The former is nonsense, the latter inconvenient. The former is nonsense because the intensity is a flux density and only defined for finite collecting areas. Thus one needs to settle on some value of the *resolution*. The latter is inconvenient because the pixels ideally are (much) smaller than the size of one's operators (e.g., an “edge detector”). I will assume that a resolution has been decided upon and that it is much larger than the pixel size. Then any pixelation is irrelevant, a mere matter of implementation (e.g., of one's printer: you never hope to see the pixelation).

The formally correct way to deal with resolution is to consider the image a member of a linear scale space. This allows changes of resolution—which are often necessary or desirable—to be defined in a principled way. In this setting the “points” of the image plane are operators that when queried yield the intensity “at that point”. These operators are “points” in the sense of “Euclid's Elements”: “A point is that which has no parts”. You cannot look *into* a point. I will refer to the size of the points as the “inner scale” of the image.

An advantage of the linear scale space setting is that it allows one to introduce partial spatial derivatives in a principled manner. One may actually take the derivative of a point and use the result as an operator that when applied to the image yields the value of the derivative at that point. This avoids the problem that images are not differentiable functions (in fact, not even functions to begin with) and that the approximate numerical differentiation of actual signals is a tricky business at best. (As people say “numerical differentiation is ill posed”.) In fact, one may find derivatives of any order of the image *at any given scale*. Whether such (perfectly good!) derivatives are actually *relevant* or *useful* depends upon the current context.

### 1.2.2.1 The Structure of Image Space

The arena of images is a fiberbundle  $\mathbb{E}^2 \times \mathbb{A}$ . Can one identify additional structure? This is of appreciable potential interest as uses of popular applications like Adobe’s Photoshop<sup>©</sup> indicate. People are ready to do all kinds of things to images, in many cases claiming to merely “improve” their image, not essentially changing it. One speaks of “straight photography”. The transformation admitted in the practice of straight photography apparently play a role not unlike “congruences” or “similarities” and one would like to relate them to the structure of image space.

At first blush one identifies similarities of the image plane and translations along the intensity axis as obvious candidates. Another, perhaps less immediately obvious group of transformations are the similarities of the intensity domain. They correspond to the well known “gamma transformations”

$$I' = I_{\max} \left( \frac{I}{I_{\max}} \right)^{\gamma}, \quad (1.25)$$

of intensities in the range  $(0, I_{\max})$ .

Next consider transformations that leave the image plane invariant but depend on both space and intensity. Here one meets with an obvious constraint. For instance, I consider the “transformation”

$$x' = J, \quad (1.26)$$

$$y' = y, \quad (1.27)$$

$$J' = x, \quad (1.28)$$

as definitely not allowable. Why? Because the image plane dimensions and the intensity dimension are mutually incommensurable. This transformation violates the condition that images are cross-sections of the fiberbundle  $\mathbb{E}^2 \times \mathbb{A}$ . On the other hand the transformation

$$x' = x, \quad (1.29)$$

$$y' = y, \quad (1.30)$$

$$J' = ax + J, \quad (1.31)$$

does not represent any problem. The factor  $a$  is apparently a *gradient*, so much intensity per unit distance. The fiberbundle structure is not violated.

The situation should look familiar to the reader of this chapter. Images are manipulated by “moving intensities” over the copies of  $\mathbb{A}$  at any point of the image plane, like beads on a string. Congruences should look like Euclidean motion in the image plane and leave distances between “beads on a single string” invariant, similarities should scale them by the same factor. This is exactly what is achieved by the group of similarities of a Cayley–Klein space with single isotropic dimension. In this case the isotropic dimension is the intensity domain. Thus I simply set:

$$x' = a(+x \cos \beta + y \sin \beta) + c_x, \quad (1.32)$$

$$y' = a(-x \sin \beta + y \cos \beta) + c_y, \quad (1.33)$$

$$J' = f_x x + f_y y + g J + h, \quad (1.34)$$

as it does precisely the right things. The subgroup that leaves the image plane invariant is evidently the most interesting. It is

$$x' = x, \quad (1.35)$$

$$y' = y, \quad (1.36)$$

$$J' = f_x x + f_y y + g J + h. \quad (1.37)$$

The parameter  $h$  controls overall brightness, whereas parameter  $g$  implements the gamma–transformations (usually denoted “contrast control”). The parameters  $\{f_x, f_y\}$  are often applied by landscape photographers as “grad filters”.

These transformations have many applications in vision. For instance, consider the local image structure

$$J(x, y) = a_{00} + (a_{10}x + a_{01}y) + \frac{1}{2!}(a_{20}x^2 + 2a_{11}xy + a_{02}y^2) + \dots \quad (1.38)$$

Using a congruency of image space it can be transformed into canonical form

$$J'(u, v) = \frac{1}{2!}(\kappa_1 u^2 + \kappa_2 v^2) + \dots \quad (1.39)$$

With an additional similarity one may even achieve

$$\sqrt{\frac{\kappa_1^2 + \kappa_2^2}{2}} = 1, \quad (1.40)$$

that is unit “curvedness”. The ratio  $\kappa_1/\kappa_2$  is a pure (second order) shape measure. The four coefficients of the cubic term are also interesting *cubic shape measures* because obviously differential invariants. It seems likely that such local measures are taken in the human visual system.

## 1.3 A Case Study: “Shape from Shading”

“Shape From Shading” is not exactly the biggest success of machine vision. It is not so clear that human vision is doing much better though. The issue remains undecided because the very aims of machine vision and human vision appear to be widely different. This makes shape from shading of some interest as a case study.

### 1.3.1 *The So Called “Shape from Shading Problem”*

The so called “Shape From Shading Problem” as conventionally construed is rather artificial and relies on numerous shaky, even *a priori* unlikely, occasionally even plainly wrong prior assumptions. Here is a rough outline:

An observer views a smooth, generically curved surface that is being illuminated such as to produce a pattern of light and shade. The task is to report the shape (that is the curvature landscape) of the surface. Sometimes the observer may also be asked for the nature of the illuminating beam (e.g., the spatial configuration and photometric properties of the “primary sources”).

As stated the problem is probably an impossible one to tackle, thus one lists any number of simplifying prior assumptions. Among these may be:

- the surface is smooth, no edges, no contours, no 3D texture (roughness);
- the surface is uniform, i.e., the same at all places;
- the surface is characterized by a single bidirectional reflectance distribution function (BRDF). Thus effects of translucency do not play a role;
- the BRDF is constant, in other words, the surface is Lambertian;
- the illuminating beam has a uniform cross section, it will illuminate a set of concentric spherical surfaces uniformly. (“A homocentric, collimated beam”);
- the illuminating beam will illuminate planes uniformly. (“A parallel beam”);
- the illumination is by primary sources only. I.e., there are no mutual interreflections;
- each point of the surface is illuminated by the same primary sources. I.e., there is no “vignetting” or “cast shadow”.

Some of these assumptions are mutually exclusive, others imply each other. Accepting some may have strong consequences, e.g., the absence of interreflections implies that the surface is either black or non-concave; the Lambertian assumption implies that the viewing geometry is irrelevant.

It is possible to frame certain limiting cases in which some of the assumptions are automatically (though approximately) satisfied. One interesting example is to assume “low relief”. This automatically takes care of the vignetting and interreflection issues. Whether such limiting cases are of any interest depend on one’s goals. If the goal is applications the assumption of low relief is likely to be frequently violated. If the goal is theoretical understanding such a limiting case may be of considerable use.



**Fig. 1.1** The Asam house at Munich. Illumination by the overcast sky from above. The material is whitewashed stucco, roughly Lambertian. There are various regions of low relief where our simplifying assumptions hold reasonably well (the clock face, the sitting putto) though there are also parts that are modeled “in the round” and where effects of vignetting and interreflection are evident. In cases like this frontal viewing is a natural condition (I made the photograph from the opposite side of the street).

A special case that appears rather limiting, yet is often applicable is that of frontal viewing (see Figure 1.1). Especially when combined with the low relief assumption this often applies to cases of real life importance, just think of viewing bas relief murals.

Another special case is that of frontal illumination. This case is completely different from frontal viewing. Moreover, it has few other applications than photographs taken with flash on the camera. Although this situation is avoided like the plague by professional photographers (it “flattens” the scene, thus works against regular shape from shading for the human observer), it is (perhaps perversely) a case for which dedicated computer algorithms have been designed.

Some of the most useful assumptions are almost certainly wrong. A key example is the assumption of Lambertian surfaces. It is a highly desirable assumption because the influence of viewing geometry vanishes, thus greatly simplifying the problem. But from physics one knows that Lambertian surfaces don’t exist. It is not that basic physics forbids them, but they can be only approximately produced even

under laboratory conditions. The non-Lambertian character of surface scattering becomes especially obvious for very oblique viewing and/or illumination directions.

Here I will assume Lambertian surfaces, low relief, frontal viewing and parallel, collimated illumination. In this case most of the usual assumptions apply (no vignetting, no interreflections, ...), even the Lambertian assumption is not problematic. This is easily the simplest setting imaginable that still holds some interest.

### 1.3.2 Setting up the Problem

Consider a relief

$$z(x, y) = z_0 + \mu w(x, y), \quad (1.41)$$

where  $\mu$  keeps track of the depth of relief. It is a convenient parameter because we simply carry calculation to 1<sup>st</sup>-order in  $\mu$ . Here is an example, the surface normals are

$$\mathbf{n}(x, y) = -\mu \left( \frac{\partial w(x, y)}{\partial x} \mathbf{e}_x + \frac{\partial w(x, y)}{\partial y} \mathbf{e}_y \right) + \mathbf{e}_z + \mathbf{O}[\mu]^2. \quad (1.42)$$

Here we obtained a significant gain in simplicity because the usual normalization factor affects only 2<sup>nd</sup> and higher orders in  $\mu$  and thus can be ignored.

I will set

$$\frac{\partial z(x, y)}{\partial x} \Big|_{x=y=0} = \frac{\partial z(x, y)}{\partial y} \Big|_{x=y=0} = 0, \quad (1.43)$$

throughout the computation because of the assumption of frontal viewing.

Assume the direction of the illuminating beam is  $\mathbf{i}$  and that it causes a normal illumination  $E_0$ . Then Lambert's Cosine Law yields the illumination pattern:

$$E(x, y) = E_0 \mathbf{i} \cdot \mathbf{n}(x, y) = E_0 \left( -\mu \left( i_x \frac{\partial w(x, y)}{\partial x} + i_y \frac{\partial w(x, y)}{\partial y} \right) + i_z \right) + \mathbf{O}[\mu]^2. \quad (1.44)$$

Notice that the absolute value of the illuminance is irrelevant. The visual system will merely record the *spatial contrast*  $C(x, y)$ , which is

$$C(x, y) = \frac{E(x, y) - E(0, 0)}{E(0, 0)} = -\frac{\mu \left( i_x \frac{\partial w(x, y)}{\partial x} + i_y \frac{\partial w(x, y)}{\partial y} \right)}{i_z} + \mathbf{O}[\mu]^2. \quad (1.45)$$

Writing

$$\mathbf{i} = -(\cos \vartheta (\cos \varphi \mathbf{e}_x + \sin \varphi \mathbf{e}_y) + \sin \vartheta \mathbf{e}_z), \quad (1.46)$$

where  $\vartheta$  denotes the elevation and  $\varphi$  the direction of the illumination, we finally obtain

$$C(x, y) = \mu \cot \vartheta (\cos \varphi \frac{\partial w(x, y)}{\partial x} + \sin \varphi \frac{\partial w(x, y)}{\partial y}) + \mathbf{O}[\mu]^2. \quad (1.47)$$

What can be observed locally is the contrast gradient  $\nabla C = C_x \mathbf{e}_x + C_y \mathbf{e}_y$ , it can be found by straight differentiation. The differentiation will generate second order derivatives of the height  $z(x, y)$ . Dropping higher order terms in  $\mu$  you obtain

$$C_x = \mu \cot \vartheta (\cos \varphi \frac{\partial^2 w(x, y)}{\partial x^2} + \sin \varphi \frac{\partial^2 w(x, y)}{\partial x \partial y}) \quad (1.48)$$

$$C_y = \mu \cot \vartheta (\cos \varphi \frac{\partial^2 w(x, y)}{\partial x \partial y} + \sin \varphi \frac{\partial^2 w(x, y)}{\partial y^2}). \quad (1.49)$$

In the “Shape From Shading Problem” the “unknowns” are  $\vartheta$ ,  $\varphi$ , and the three 2<sup>nd</sup>-order partial derivatives of the height of relief  $z(x, y)$ . For this we have two equations, the observables  $C_x$  and  $C_y$ . The problem is evidently underdetermined, even in this simplest setting.

One ambiguity that is clearly unavoidable is the mix up between the height of contrast and the elevation of the source as expressed through the factor  $\mu \cot \vartheta$ . A scaling  $\mu w(x, y)$  can be undone by adjusting  $\vartheta$ . We may as well notice this relation and proceed to eliminate  $\vartheta$ , obtaining a homogeneous equation for the three partial derivatives.

Writing  $\nabla C = G(\cos \gamma \mathbf{e}_x + \sin \gamma \mathbf{e}_y)$  we obtain

$$\sin \gamma \cos \varphi z_{xx} - \cos(\gamma + \varphi) z_{xy} - \cos \gamma \sin \varphi z_{yy} = 0, \quad (1.50)$$

where I have introduced a more concise notation for the partial derivatives. Apart from this we have that the height of relief is undefined. This may be expressed through the equation

$$\frac{1}{2}(z_{xx}^2 + 2z_{xy}^2 + z_{yy}^2) = \text{constant}, \quad (1.51)$$

the height being absorbed in the elevation of the source. (The expression is the “curvedness”, see below.)

Thus we end up with one parameter (the elevation of the source) remaining fully unspecified and two equations for four unknowns (the illumination direction and three partial derivatives of the height). It would help to know the direction of illumination, but even then we still have only two equations for the three partial derivatives. The problem is evidently very underspecified.

**Ways to proceed.** There are various ways to proceed from here. Well known methods from machine vision recognize the fact that the local conditions are insufficient to find the local shape (curvature) and reformulate the Shape From Shading Problem into a global problem, using either partial differential equations or a variational method. Thus one introduces surface integrity constraints to force a solution. These methods are well known and I will not pursue them here because they are quite unlike anything that might be attempted by the human visual system. I will stubbornly pursue the purely local problem in an attempt to guess what the visual system might be doing.

**Posing the local problem in a more symmetrical way: describing 2<sup>nd</sup>-order shape.** The description of 2<sup>nd</sup>-order surface shape in terms of partial derivatives in a Cartesian frame in the tangent plane is often convenient, but masks the symmetries of the 2<sup>nd</sup>-order structure. Here is a better adapted description:

Notice that a term like  $x^2 + y^2$  is rotationally symmetric whereas terms like  $xy$  and  $x^2 - y^2$  have two lines of bilateral symmetry. The latter two terms are very similar and can be transformed into each other through a rotation of the coordinate system over  $\pi/4$ . Hence the transformation

$$z_{xx} = r + t, \quad (1.52)$$

$$z_{xy} = s, \quad (1.53)$$

$$z_{yy} = t - r, \quad (1.54)$$

thus we obtain

$$z(x, y) = \frac{1}{2}(z_{xx}x^2 + 2z_{xy}xy + z_{yy}y^2) = r \frac{x^2 - y^2}{2} + sxy + t \frac{x^2 + y^2}{2}. \quad (1.55)$$

I treat the surface as in visual space, that is to say, differential invariants like the mean and Gaussian curvature are calculated as in singly isotropic (the z-direction) space. This meshes perfectly with the assumption of “low relief”. (This even allows one to introduce an overall surface slant without any complication.)

The principal curvatures are

$$\kappa_{1,2} = t \pm \sqrt{r^2 + s^2}, \quad (1.56)$$

and the principal directions are

$$\{r \pm \sqrt{r^2 + s^2}, s\}. \quad (1.57)$$

Thus the mean curvature  $H$  and Gaussian curvature  $K$  are

$$H = \frac{1}{2}(\kappa_1 + \kappa_2) = t, \quad (1.58)$$

$$K = \kappa_1 \kappa_2 = -r^2 - s^2 + t^2. \quad (1.59)$$

The expression  $\frac{1}{2}(\kappa_1 - \kappa_2)$  may be called the “non-sphericity” as it measures the deviation from rotational symmetry. It equals  $\sqrt{r^2 + s^2}$ , thus sphericity implies  $r = s = 0$ .

The “curvedness”  $\chi = \sqrt{\frac{1}{2}(\kappa_1^2 + \kappa_2^2)}$  measures the deviation from planarity and turns out to be  $\sqrt{r^2 + s^2 + t^2}$ . The “shape index” specifies the pure shape and is defined as

$$\sigma = \arctan \frac{\kappa_1 + \kappa_2}{\kappa_1 - \kappa_2} = \arctan \frac{t}{\sqrt{r^2 + s^2}}. \quad (1.60)$$

The shape index takes values on  $[-\frac{\pi}{2}, +\frac{\pi}{2}]$ .

All this can be summarized in an intuitively very attractive manner. The space of all 2<sup>nd</sup>-order surface shapes is best represented by a Cartesian  $\{r, s, t\}$ -space. The origin represents planarity, i.e., shapelessness, whereas distance from the origin, the “curvedness” implies deviation from the tangent plane. On the surface of the unit sphere the latitude is the shape index, whereas the longitude indicates twice of the direction of principal curvature. The natural representation is in polar coordinates

$$r = \chi \cos \sigma \cos \psi, \quad (1.61)$$

$$s = \chi \cos \sigma \sin \psi, \quad (1.62)$$

$$t = \chi \sin \sigma. \quad (1.63)$$

This direction of principal curvature (that is  $\frac{\psi}{2}$ ) is undefined at the poles because the  $t$ -axis represents the spherical shapes. Notice that antipodes are mutually related as a cast and its mold.

### 1.3.3 The Local Shape from Shading Problem

The local Shape From Shading problem is best recast in terms of the symmetrical parameters  $\{r, s, t\}$  introduced above.

The ambiguity due to the elevation of the illumination means that shape inferences have to be done *modulo* the curvedness, which again means that the space of possible inferences is reduced to the lines through the origin of shape space, a projective plane. We may represent it by the unit sphere in shape space with pairs of antipodal points identified.

The observation of the contrast gradient yields the constraint

$$r \sin(\gamma + \varphi) - s \cos(\gamma + \varphi) + t \sin(\gamma - \varphi) = 0, \quad (1.64)$$

which is a homogeneous, linear equation thus a plane through the origin of shape space, which meets the unit sphere in a great circle. At this point we may simplify the expression by specializing the coordinate system, letting the first frame vector coincide with illuminance surface flow direction. Thus, setting  $\gamma \rightarrow 0$  the constraint is

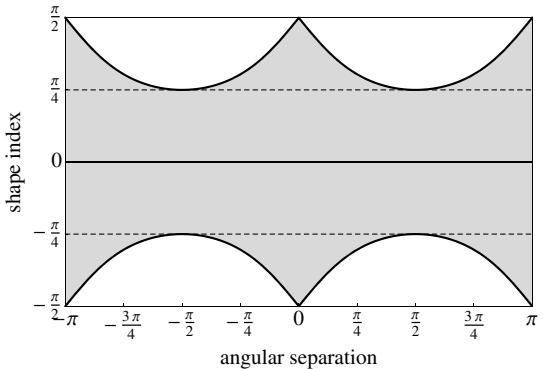
$$r \sin \varphi - s \cos \varphi - t \sin \varphi = 0. \quad (1.65)$$

The pole of this great circle is

$$\mathbf{p}(\varphi) = \frac{\sin \varphi \mathbf{e}_1 - \cos \varphi \mathbf{e}_2 - \sin \varphi \mathbf{e}_3}{\sqrt{1 + \sin^2 \varphi}}. \quad (1.66)$$

Of major interest is the colatitude of the pole, because it specifies the extreme values of the possible shape index inference. It is

**Fig. 1.2** The range of feasible shape indices (dark) as a function of the angular separation between the direction of illumination and the direction of the contrast gradient.



$$\sigma_{\max} = \frac{\pi}{2} - \left| \arcsin \frac{\sin \varphi}{\sqrt{1 + \sin^2 \varphi}} \right|. \quad (1.67)$$

Although it is quite possible to infer any hyperbolic shape, no matter what the value of  $\varphi$  might be, there is a maximum to the shape index of elliptic inferences (see Fig. 1.2). For instance, a spherical inference implies  $\gamma = \varphi$  (of course modulo  $\pi$ ).

### 1.3.3.1 The “Observables” for the Shape from Shading Problem

In the literature on the Shape From Shading Problem the generic assumption is that the relevant observable is the spatial contrast. (For the local problem this reduces to the contrast gradient, though this plays no role in the machine vision literature.) However, this assumption may well be questioned.

Various attempts to find the illuminance direction from the image are found in the literature, most of them ad hoc, some of them mere shots in the dark.

A principled manner to find the illuminance direction from the image is available if the surface is corrugated such as to yield a visible illuminance induced texture. Such a method works if the statistical structure of the corrugations is isotropic. The basic idea is simple enough. An isotropic protrusion will yield a dipole pattern, light on the side facing the source, dark on the side facing the other direction. An isotropic indentation will also yield a dipole pattern, it will have the same axis as that of the protrusion, but the opposite polarity. Thus the gradients have the same *orientation*, but opposite *directions*. The average gradient of an isotropic texture will indeed tend to zero, but the average *squared* gradient will have the correct orientation. This is the crux of the structure tensor method. One computes the eigensystem of the “structure tensor”

$$S = \langle \nabla C^\dagger \cdot \nabla C \rangle = \begin{pmatrix} \langle C_x C_x \rangle & \langle C_x C_y \rangle \\ \langle C_y C_x \rangle & \langle C_y C_y \rangle \end{pmatrix}. \quad (1.68)$$

The direction of the largest eigenvector is the orientation of the illumination flow [8].

This method has been shown to work very well with a large variety of 3D textures. Isotropy is essentially the only requirement. It has also been shown that the human observer uses this method and typically finds the orientation with an accuracy of about 5°.

### 1.3.3.2 Human Vision and Shape from Shading

We have solved the Local Shape From Shading Problem above and we have introduced the possibility of additional observational evidence. This should be sufficient to investigate the angle human vision takes on the problem.

We identify three cases:

- the observer lacks any prior information, except for the general setting: the observer is looking at a frontoparallel, Lambertian (e.g., plaster or marble) plane with low relief modulation, illuminated by a uniform, parallel, collimated beam (e.g., the sun);
- the observer additionally has prior knowledge concerning the illumination direction (e.g., through observation of 3D texture induced contrast);
- the observer has prior information concerning the shape (e.g., knows it to be spherical).

In the first case the observer is supposed to estimate both the shape and the illumination direction, in the second case only the shape and in the third case only the direction of illumination. The first case is the most interesting, though especially the second case may be expected to have frequent application. An overview of the various relations is graphically illustrated in figure I.3.

Consider the first case. The observer observes the direction of the contrast gradient, we specialize the coordinate system such that  $\mathbf{e}_x$  is in the contrast gradient direction. Of course there remains a  $\pm\pi$  ambiguity here. Then we know that the shape index is limited as

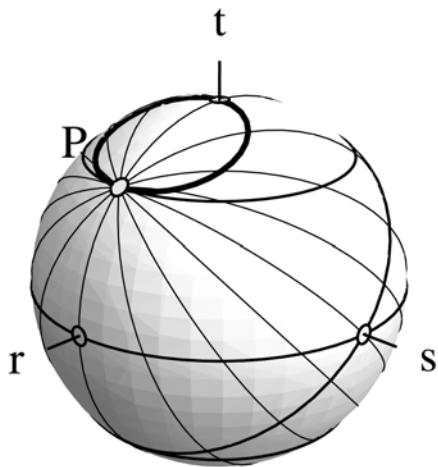
$$\sigma \leq \frac{\pi}{2} - \left| \arcsin \frac{\sin \varphi}{\sqrt{1 + \sin^2 \varphi}} \right|, \quad (1.69)$$

where the direction of illumination  $\varphi$  is supposed to be fully unknown.

It is *always* possible to infer a symmetrical saddle ( $s = 0$ ), for *any* direction of illumination. It is also possible to infer a spherical shape ( $s = \pm\frac{\pi}{2}$ ), though this implies that the illumination direction coincides with the contrast gradient direction ( $\varphi = 0$ ), a very specific condition. Of course the saddle would have to be in a specific orientation, whereas the sphere looks the same in *all* orientations. All considering, it is hard to make a principled choice. Of course *any* shape is possible, with more complicated constraints on the directions.

Consider the second case. If the direction of illumination is known, there is a constricted range of possible values of the shape index. Granted a preference for elliptical shapes (see below) one expects the system to select the largest possible value, then the best guess is

**Fig. 1.3** The unit sphere in rst-space (equation I.51), where the r-dimension is the direction of the relative contrast gradient. The sheave of great circles through P (mid-point of the arc rt) are the constraint planes (equation I.50) for the various illumination directions (the poles lie on the great circle passing through s.) The fat small circle is the locus of “most spherical inferences”. At P the inference is cylindrical and all illumination directions go, at t the inference is spherical and the illumination direction coincides with the gradient direction.



$$\sigma = \frac{\pi}{2} - \left| \arcsin \left( \frac{\sin \varphi}{\sqrt{1 + \sin^2 \varphi}} \right) \right|. \quad (1.70)$$

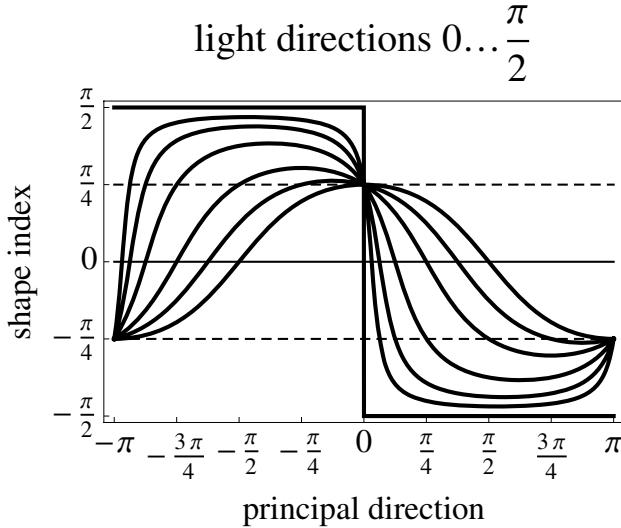
Thus knowledge of the illumination direction fails to nail the shape, but does constrain the possibilities. (See Figure I.4.)

Finally consider the third case. Knowing the shape means that only points on the latitude circle of that shape are feasible. Thus the solutions lie on the intersection of the great circle defined by the constraint and this latitude (small) circle. It is possible that no solution exists, otherwise there are two distinct ones. If so, the solution is

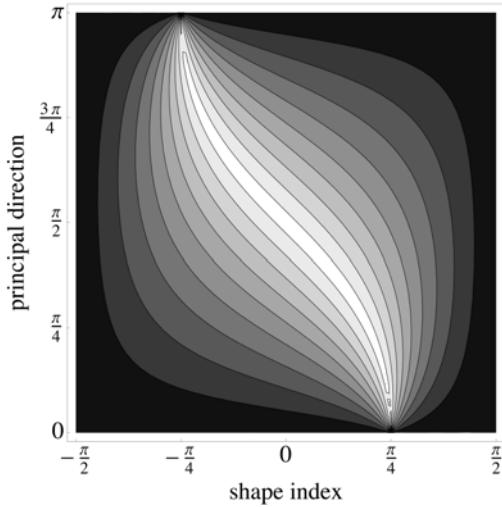
$$\varphi = \arccos \frac{|\sin \sigma - \cos \psi \cos \sigma|}{\sqrt{1 - \cos 2\sigma \cos \psi}}. \quad (1.71)$$

In this case one might run into a *contradiction*, which is the strongest constraint possible. Otherwise such a prior knowledge pretty much nails the direction of illumination (see Figure I.5). A key example is the spherical shape which may act like a “wind sack” for the flow of light.

**How well is the human visual system doing?** In the absence of prior knowledge it appears to be the case that the human observer invariably reports a spherical shape. Apparently the visual system considers the spherical inference the best bet. This may be due to the fact that (overall) smooth objects are likely to be predominantly convex. The remaining convex/concave ambiguity is usually resolved by the prior assumptions that illumination tends to be from above. If the illumination is actually



**Fig. 1.4** Assume the direction of illumination is known (each curve is for a specific direction of illumination), then the shape (as specified by the shape index) still depends upon the (unknown) direction of principal curvature. Notice that an elliptical inference (convex or concave) is always possible, though generically not spherical.

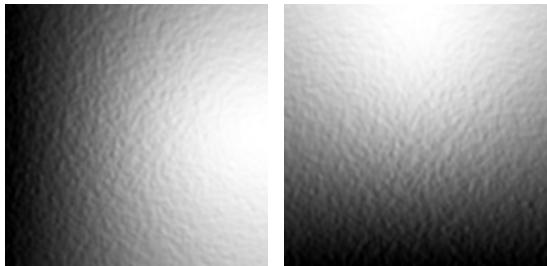


**Fig. 1.5** Suppose the shape (specified by the shape index) is known. If the direction of principal curvature is also known the light direction is fully determined, otherwise there exists a one-parameter ambiguity. The shades run from black = 0 to white =  $\frac{\pi}{2}$ .

from the side humans typically report convexities, thus an additional bias seems to be on convexity as opposed to concavity.

Notice that these “inferences” are in fact “hallucinations”. In reality any shape goes! Yet human observers rarely feel that they are *guessing*, the experience is that of *seeing* a specific shape.

**Fig. 1.6** Two renderings of a rough surface illuminated from the right. Both surfaces are quadrics viewed frontally at the center of the figure. The surface at right is spherically convex, that at left is a symmetric saddle. Notice that the 3D–texture *visually* reveals the direction of illumination.



If the direction of illumination is clearly visible one expects the human observer to use this. (It has been established that human observers indeed see the illumination direction based on 3D texture.) Perhaps surprisingly, *they don't*. People tend to report sphericity, even when there is no elliptic solution at all, that is when the contrast gradient is orthogonal to the illumination direction! (see Figure 1.6) Thus there can be no doubt that a machine algorithm would do better. What frequently happens in such cases is that the presentation “splits” into two layers (like it does often as in cases of apparent transparency). One layer has the 3D texture, illuminated veridically whereas the other layer gets the (spherical) shape, illuminated from a direction perpendicular to the (clearly visible!) veridical direction.

We conclude that the visual system attempts to solve the local problem (where machine vision gives up), but doesn't do too well on it. One could easily beat it with a simple machine algorithm.

**Remaining questions.** As always in the study of human visual perception, many questions remain. One is whether the visual system does anything global in addition to purely local inferences. Whereas it is very unlikely that the system does anything remotely like the current machine algorithms, there remain a number of intriguing possibilities.

It is likely that the visual system pursues “local” inferences on various levels of resolution. Whereas this would hardly be of much interest if there were no “added value” to it, this would be of interest because the assumption that the illumination directions are the same irrespective of the level of resolution is a very reasonable one. For articulated surfaces (a globally quadric surface would not profit at all) this is a very promising proposition. Such methods would be in between local and global, though quite different from the global algorithms in use by the machine vision community today.

It is also likely that the system would not stop at the 2<sup>nd</sup>-order surface structure. There is much reason to believe that the 3<sup>rd</sup>-order surface structure has to be very relevant for shading based inferences. For instance, the singular points of the illuminance pattern (extrema and saddle points) occur at the parabolic points of the surface [Q]. At such points the surface is locally degenerate (cylindrical) and for a generic description one has to take the cubic terms into account. The cubic terms will also affect the Hessian (in addition to the gradient) of the contrast and it is very

likely that the visual system is sensitive to those, indeed, perhaps more so than to the gradient. As we have argued above, at any given point the gradient can be transformed away through a congruence in image space. Little is known about the way cubic structure appears in shading, although the potential importance cannot be in doubt.

## 1.4 Final Remarks

“Final remarks” is more apt than “conclusions” at this point, because there isn’t any real “conclusion”. What I have tried to convey in this chapter is the enormous gap between the attempts to understand the functioning of the human visual system in a formal way and the attempts to implement machine systems that “see”. One might expect these endeavors to overlap appreciably because both the initial “data” and the final “goals” are very similar or even identical. Moreover, the generic human observer and the generic machine vision system are supposed to function in very similar (often identical) worlds. However, the differences are very significant. They are mainly due to hardware constraints. The major bottlenecks of biological as opposed to artificial systems are:

- whereas absolute calibrations or at least fixed operation points are usually no problem in artificial systems, such luxuries are not available in biological systems. In biological systems any level has to dynamically shift its operation level in order to keep signals within the (very limited) dynamical range and these levels are not known to other parts of the system (or even the individual subsystem itself);
- in biological systems local processing is the rule, global processing only works in very coarse grained (sub-)systems (that is to say, low resolution is substituted as a cheap replacement of true globality). This rules out most algorithms that have made machine vision into a viable technology.

As opposed to these limitations biological systems also have major strengths, the main one being the full integration of the “background”. Biological systems are part of their biotopes and background knowledge of the structure of the biotope is evident at all levels of implementation, from the optics of the eye to the nature of the “hallucinations”. This is an aspect that machine vision has hardly touched upon.

The examples I gave in this chapter I believe to be typical in showing up such differences.

The limitations of biological systems may be a burden to an engineer designer, but evolution has done remarkably well given these constraints. Thus I believe that machine vision has something to learn from biological implementations even though I also believe that biological systems are bound to be beaten in many subdomains by well designed artificial systems, certainly on the long run. The lessons will be (of course) very general design principles and in this chapter I have tried to outline a few.

Areas where the human visual system is unlikely to give way to machine implementations are those of the visual arts. However, such achievements are very

difficult to measure up. There are essentially no yardsticks for the products of the creative arts. The only way to assess this is to try to find whether there is a market (in the art galleries circuit) for machine generated products. If human artists are eventually muscled out of these circuits, machines will finally be in power. That'll be the day.

## References

1. <http://www.research.ibm.com/deepblue/>
2. Koenderink, J.J.: The brain a geometry engine. Psychological Res. 52, 22–127 (1990)
3. [http://nl.wikipedia.org/wiki/Sherlock\\_Holmes](http://nl.wikipedia.org/wiki/Sherlock_Holmes)
4. Lotze, R.H.: Medicinische Psycholohie oder Physiologie der Seele. Weidman'sche Buchhandlung, Leipzig (1852)
5. Riemann, B.: Über die Hypothesen, welche der Geometrie zu Grunde liegen. Habilitationsschrift, Abhandlungen der Kniglichen Gesellschaft der Wissenschaften zu Gttingen 13 (1854)
6. Sachs, H.: Ebene isotrope Geometrie. Vieweg-Verlag, Wiesbaden (1987)
7. Yaglom, I.M.: A simple non-Euclidean geometry and its physical basis: an elementary account of Galilean geometry and the Galilean principle of relativity (Translated from the Russian by Abe Shenitzer). Springer, New York (1997)
8. Koenderink, J.J., Pont., S.C.: Irradiation direction from texture. J. Opt. Soc. Am. A 20, 1875–1882 (2003)
9. Koenderink, J.J.: Photometric invariants related to solid shape. Optica. Acta. 27, 981–996 (1980)

## Chapter 2

# Knowing a Good Feature When You See It: Ground Truth and Methodology to Evaluate Local Features for Recognition

Andrea Vedaldi, Haibin Ling, and Stefano Soatto

**Abstract.** While the majority of computer vision systems are based on representing images by local features, the design of the latter has been so far mostly empirical. In this Chapter we propose to tie the design of local features to their systematic evaluation on a realistic ground-truthed dataset. We propose a novel mathematical characterisation of the co-variance properties of the features which accounts for deviation from the usual idealised image affine (de)formation model. We propose novel metrics to evaluate the features and we show how these can be used to automatically design improved features.

## 2.1 Introduction

Local features are the building blocks of many visual recognition systems. They are deterministic functions of the image (i.e., statistics) designed to minimize the effect of various “nuisance factors” such as illumination and viewpoint, while at the same time remaining representative of the object or category at hand.

Local features are typically designed by exploiting common sense, sometime drawing inspiration from current knowledge of the human visual system, without a direct tie to the task at hand. So, we cannot say that any of the existing features is the best possible one could design for the specific task of recognition. And it could not be otherwise. Elementary decision-theoretic considerations reveal that the best

---

Andrea Vedaldi

University of California at Los Angeles, Los Angeles, USA

e-mail: [vedaldi@cs.ucla.edu](mailto:vedaldi@cs.ucla.edu)

Haibin Ling

Temple University, Philadelphia, USA

e-mail: [hbling@temple.edu](mailto:hbling@temple.edu)

Stefano Soatto

University of California at Los Angeles, Los Angeles, USA

e-mail: [soatto@cs.ucla.edu](mailto:soatto@cs.ucla.edu)

possible feature is the trivial one – the image itself – as no deterministic function of the data can “create information,” even without getting into too much detail on what “information” means in the context of visual recognition.

So why would anyone want to use local features, let alone design or compare them? For one, they seem to work, and it is worthwhile trying to understand why and how.<sup>1</sup> Given that we are not going to design features for optimality in the end-to-end task, can we at least *test their effectiveness*? How do we compare two features? How can we say that one is better than the other?

So far, all comparisons of local features have been *empirical*. That is, their effectiveness is measured by recognition performance in an end-to-end task, where the features are one element of the decision process, together with the classifier and the dataset. An empirical test can tell which one is the better feature among the group being tested, but it tells us nothing on how a given feature can be improved, or how performance generalizes to different classifiers and different data sets.

In this Chapter we introduce a different methodology for evaluating features. We call this *rational evaluation*, as opposed to empirical, even though it naturally entails an experiment.

The first thing we need is *ground truth*. If features were designed for optimality in an end-to-end task (in which case they would have to be co-designed with the classifier), then any labeled training set, along with standard decision-theoretic tools, would suffice. But features are not co-designed with the classifier, so they should be evaluated independently of it. For that we need ground truth. In this Chapter we describe a way to design ground-truthed data to evaluate the effectiveness of a given feature based on its underlying (explicit or implicit) invariance assumptions. Such data consists of *synthetic images*, generated with a model that strictly includes the model underlying the invariance assumptions of a given feature. While ultimately an end-to-end system should be evaluated on the recognition task performed on real images, there is no straightforward way to distill the role of features unless proper ground truth is available.

Once we have ground truth, we need to elucidate the various components of the feature design process, that includes a choice of image domain (the “feature detector”), a choice of image statistic computed on such a domain (the “feature descriptor”), and a choice of decision function (“feature matching”) that becomes the elementary tool of the classifier downstream.

The effect of this procedure is not just a number to rank existing features based on how well they perform, when coupled with a given classifier, on a given dataset. A rational comparison also provides ways to improve the design of the feature, as we illustrate with an example. A similar approach could be followed to design better descriptors, and also better detector.

This Chapter is part of a three-prong approach we have been developing for designing and evaluating local features: In [15] we provide a reliable open-source

---

<sup>1</sup> Even though, theoretically, one could “learn away” nuisances with a super-classifier that would take the raw images as input, such a classifier may be too hard to design, or require too much data to train, especially for adversarial nuisances such as occlusions.

implementation of some of the most common local features. In this manuscript we describe a methodology to compare local features. Finally, in [14] we provide code to generate synthetic test images, as well as a number of already rendered samples.

## 2.2 Empirical Studies of Local Features

Because of their prominent role in recognition systems, local features have been the subject of considerable attention in the Computer Vision community. Due to the difficulty of extracting adequate ground truth, however, direct evaluation (i.e., not part of an end-to-end system) has been mostly limited to planar scenes [2] designed to fit the conditions for which the features were designed. While local features are usually designed to be invariant to a simple class of transformations (say affine, or projective, corresponding to the assumption of planar scenes), it is the behavior of the feature in the presence of violations of such assumptions that determines its effectiveness. Therefore, it is important that the ground truth reflects conditions that supersede the underlying assumptions.

The need to test features on more challenging data has driven some to employ synthetic datasets [12, 5], although the resulting images lacked in visual realism. More realistic data was used by [2] to infer ground truth via stereo. This procedure, however, is difficult to scale up to be representative of the complexity and variability of natural scenes. The most extensive collection of real objects to-date is [10], where a selection of (uncluttered) objects was placed on a calibrated turntable in front of a blue screen. Thousands of features were mapped from small-baseline image pairs to wide-baseline views in a semi-automated fashion. A semi-synthetic data set was produced in [13] by gathering range images acquired with a laser scanner and generating a number of artificial views by rotating the data. [17] recognized the importance of obtaining wide-baseline feature deformation data for the study of viewpoint-invariant features and used structure from motion to estimate re-projection of point features from a large number of views of real scenes. Unfortunately this technique provides only limited ground truth information (i.e., sparse 3-D points estimated from the images themselves) and is laborious to collect, especially for controlled experimental conditions. To this date, however, there is no extensive data set that can scale up to an arbitrarily large number of scenes, where the geometry of the scene, its reflectance, the illumination, sensor resolution, clutter, and lens artifacts can be controlled and analyzed by the user.

In order to make a useful tool for evaluating features, however, it is not sufficient to generate (even a lot of) synthetic scenes with ground truth. We have to develop a methodology that allows us to evaluate different aspects of the feature matching process in isolation if we want to rationally improve the design of existing features. The following section does just that. While the nomenclature we introduce may seem like a burden to the reader at first, it will make the evaluation process more rigorous and unequivocal.

### 2.2.1 Some Nomenclature

The image is a function from a domain (the lattice, or the real plane) to a range (the positive reals, possibly quantized into levels, or the three color channels). A *local feature* is a local image statistic. That is, a deterministic function of the image restricted to a neighborhood. A neighborhood is a compact, simply connected subset of the image domain, which is often referred to as a “*region*.” A local feature that does not depend on a particular parameter or function is said to be *invariant* to it. A desirable feature is one that is invariant to phenomena that are independent of the identity of the object or category of interest, often called *nuisance factors*. A feature is called *distinctive* if, when considered as a function of the object or category of interest, it is not a constant. A desirable feature is one that provides a “signature” of the object of interest. We focus our attention on the two most common nuisance factors, illumination and viewpoint, and seek for features that are distinctive of the shape and reflectance properties of the object or category of interest. Conceptually, the design of such features can be broken down into steps:

**Detection.** Given an image, the *co-variant detector*, or simply “detector”, selects a number of image regions. It is designed to extract the same (deformed) regions as the image deforms under a viewpoint change. A specific detector (SIFT, Harris-Laplace, Harris-Affine) is compatible by design with a certain family of such deformations (usually a group, e.g., similarities, affinities [9]). Section 2.3.1 develops a formal model of this step.

**Canonization.** The co-variant regions are *canonized*, i.e., deformed to a standard shape. This process compensates (in part or entirely) for deformations induced by the companion transformations. It is often assumed that such transformations form a group, and therefore they can be undone (inverted).

**Description.** The *descriptor* computes a statistic of the image on the canonized co-variant regions. This process may eliminate, or render the descriptor insensitive to, additional deformations which are not removed by canonization.

**Matching.** A *similarity measure* is used to compare invariant descriptors to match regions in different images.

## 2.3 Constructing a Rigorous Ground Truth

In Section 2.3.1 we introduce an idealized model of the output of co-variant detectors and in Section 2.3.2 a model of feature correspondences. These will be used in the empirical analysis in Section 2.3.3 and 2.3.4.

### 2.3.1 Modeling the Detector

Viewpoint has a direct effect on the *geometry* of local features, resulting in a deformation of their shape and appearance. The purpose of a (*co-variant*) *detector* is to select regions that warp according to, and hence track, image deformations induced by viewpoint changes.

frame	companion warps	fixed subset
point	homeomorphisms	translations
disk	similarities	translations and scalings
oriented disk	similarities	similarities
ellipse	affinities	affinities up to residual rotation
oriented ellipse	affinities	affinities

**Fig. 2.1** Feature frames. Top. The figure depicts the five classes of feature frames, together with their parameters and the selected point  $z$  used to represent orientation. From left to right: point, disk, oriented disk, ellipse, oriented ellipse. Bottom. Association of frame types to companion warps used in this Chapter.

There is a correspondence between the type of regions extracted by a detector and the deformations that it can handle. We distinguish transformations that are (i) compatible with and (ii) fixed by a detector. For instance, a detector that extracts disks is compatible with, say, similarity transformations, but is not compatible with affine transformations, because these in general map disks to other type of regions. Still, this detector does not fix a full similarity transformation, because a disk is rotationally invariant and that degree of freedom remains undetermined. These ideas are clarified and formalized by the next definitions.

**Frames.** Typically one models the output of a detector as image regions, i.e., as subsets of the image domain [2]. However, many popular detectors produce “attributed regions” instead (for example the SIFT detector [6] produces oriented disks rather than just disks). Since such attributed regions are ultimately used to specify image transformations, in this work we refer to them as “frames.” Thus a *frame* is a set  $\Omega \subset \mathbb{R}^2$  possibly augmented with a point  $z \in \Omega$ . For example, a disk is a set  $\Omega = \{|x - x_0|_2 < r\}$  and an oriented disk is the combination  $(\Omega, z)$  of a disk and a point  $z \in \Omega$ ,  $z \neq x_0$  representing its orientation<sup>2</sup> (as the line connecting the center  $x_0$  to  $z$ ). Here we consider the following classes of frames (see Figure 2.1), that capture the output of most detectors found in the literature:

- **Points.** Points are determined by their coordinates  $x_0$ .
- **Disk.** Disks are determined by their center  $x_0$  and radius  $r$ .

<sup>2</sup> We prefer to use a point  $z$  rather than specifying the orientation as a scalar parameter because this representation is easier to work with and can be easily generalized to more complex feature frames.

- **Oriented disks.** Oriented disks are determined by their center  $x_0$ , radius  $r$  and orientation  $\theta$ .
- **Ellipses.** Ellipses are determined by their center  $x_0$  and the moment of inertia (covariance) matrix

$$\Sigma = \frac{1}{\int_{\Omega} dx} \int_{\Omega} (x - x_0)(x - x_0)^{\top} dx.$$

Note that  $\Sigma$  has three free parameters.

- **Oriented ellipses.** Oriented ellipses are determined by the mapping  $A \in GL(2)$  which brings the oriented unit circle  $\Omega_c$  onto the oriented ellipse  $\Omega = A\Omega_c$ .

**Frames fix deformations.** Each type of frame (point, disk, oriented disk, etc.) can be used to fix (and undo, by canonization) certain image transformations. In fact, given a pair of frames  $\Omega_1$  and  $\Omega_2$ , the equation  $\Omega_2 = w\Omega_1$  determines (partially or entirely) the warp  $w$ . Therefore, a frame  $\Omega$  acts as a reference frame to specify deformations. This fact is captured by the following definitions:

- **Frame deformations.** For what concerns our discussion, an image deformation (warp)  $w$  is simply a transformation  $\mathbb{R}^2 \rightarrow \mathbb{R}^2$  of the image domain, and  $wI$  denotes the image  $I(w^{-1}(x))$ . Such deformations apply to frames as well: Given a frame  $(\Omega, z)$ , the warped frame  $w(\Omega, z)$  is the pair  $(w\Omega, w(z))$ . Note that, if present, the selected point  $z$  is moved too; later we will use the shorthand notation  $w\Omega$ , still meaning that the warp applies to both the set and the selected point  $z$ .
- **Closed, complete, and free frames.** Frames are *closed* under the deformations  $\mathcal{W}$  if warping a frame by  $w \in \mathcal{W}$  does not change their type. For example, disks and oriented disks are closed under similarity transformations and ellipses and oriented ellipses are closed under affine transformations. We say that a frame is *complete* for a certain set of transformation  $\mathcal{W}$  if the equation  $\Omega_2 = w\Omega_1$  admits at most one solution  $w \in \mathcal{W}$ . We also say that the frames are *free* on the set  $\mathcal{W}$  (as in “free generators”) if such an equation has a solution for all possible pairs of frames  $\Omega_1$  and  $\Omega_2$ .

When analyzing a detector, it is important to specify both the type of frames it produces and the class of transformations that are assumed, which we call *companion warps*. Notice in fact that each frame type can be used in connection with different types of transformation, so both choices must be specified. In the rest of the Chapter we focus on the most natural cases, summarized in Figure 2.1. For instance, from the table we read that disks are used in conjunction with similarity transformations (their companion warps), but are expected to fix only a subset of them.<sup>3</sup>

<sup>3</sup> Notice also that frames (i) are closed under the companion warps, (ii) complete for a subset of these, and (iii) free on the complete subset. Property (iii) is not always satisfied by real detector. For instance, maximally stable extremal regions [8] have arbitrary shape  $\Omega$ , but their companion warps are just affine transformations. This means that the equation  $\Omega_1 = w\Omega_2$  may not have a solution.

### 2.3.2 Modeling Correspondences

In the previous section we have modeled the detector as a mechanism that extracts (co-variant) frames. Operatively, the output of the detector is used to establish frame-to-frame correspondences between multiple images of the same object. For evaluation purposes, it is therefore necessary to extract sets of corresponding frames. This idea is captured by the following definitions.

**View sets (multiple views).** A *view set* [4]  $\mathcal{V}$  is a collection of images  $(I_1, \dots, I_n)$  of a scene taken under different viewpoints. Under Lambertian reflection and other assumptions [16], any image  $I_j(x)$ ,  $x \in \Lambda$  in a view set is obtained from any other image  $I_i$  by a deformation  $I_j(x) = I_i(h_{ij}(x)) \doteq (h_{ji}I_i)(x)$ . Such a deformation arises from the equation

$$h_{ij}(x) = \pi(R_{ij}\pi_j^{-1}(x) + T_{ij}), \quad x \in \Lambda \quad (2.1)$$

where  $\pi$  is the perspective projection and  $\pi_j^{-1}(x)$  is the pre-image of pixel  $x$  from viewpoint  $j$  and  $(R_{ij}, T_{ij})$  is the camera motion from view  $j$  to view  $i$ . Also note that, due to occlusions and other visibility artifacts, equations  $I_j = h_{ji}I_i$  may have only local validity, but this is sufficient for the analysis of local features.

**Co-variant frame sets (correspondences).** A (*co-variant*) *frame set*  $\mathcal{F}$  is a selection of frames  $(\Omega_1, \dots, \Omega_n)$ , one for each image of a view set  $\mathcal{V} = (I_1, \dots, I_n)$ , that are linked by the same deformations of the view set, i.e.,

$$\Omega_i = h_{ij}\Omega_j$$

where  $h_{ij}$  is given by (2.1). It is useful to think of co-variant frames as collections of geometric elements (such as points, regions, bars and so on) that are “attached” to the images and deform accordingly. Co-variant frames define the support of features and, by tracking image deformations, enable canonization.

**Frame sets enable canonization.** By mapping a co-variant frame  $\Omega_i$  to a *canonical variant*  $\Omega_0$ , the equation  $\Omega_0 = w_i\Omega_i$  defines a warp  $w_i$  which undoes the local image deformation in the sense that the local appearance  $w_iI_i$  is constant through the view set  $i = 1, \dots, n$ . For example, mapping an oriented disk  $\Omega_i$  to the disk  $\Omega_0 = w_i\Omega_i$  of unit radius and orientation  $z = (0, 1)$  undoes the effect of a similarity transformation. Doing so for an un-oriented disk does the same up to a residual rotation.

*Remark 2.1.* Operatively, a detector can attach a frame to the local appearance only if this has enough “structure:” We can associate a disc to a radially symmetric blob, but we cannot (uniquely) associate an oriented disc to it because the image is rotationally symmetric. It should be noted, however, that this is irrelevant to the end of canonization: As long as the most specific frame is attached to each image structure, canonization will make the local appearance constant. For example, we cannot associate an oriented disk to a symmetric blob, but this is irrelevant as the residual rotation does not affect the local appearance by definition.

While so far we have just listed nomenclature, the next section will tie these concepts to the empirical process of evaluating features relative to ground truth.

### 2.3.3 Ground-Truth Correspondences

The main obstacle to the practical applicability of the concept of co-variant frames given in Section 2.3.1 is that the actual image transformations  $h_{ij}$  (2.1) are rarely of the idealized types because world surfaces are seldom flat, so the actual pixel motion  $h(x)$  is more complex than a similarity or other simple transformation that we might assume. Furthermore, due to occlusion, folding, visibility and reflectance phenomena, images in a view set are rarely related to one another by simple deformations of their domains.

Therefore, we relax our requirement that the frames represent exactly the image deformations, and look for the best fit. We propose the following operational construction of a ground-truth frame set (i.e., of ground-truth correspondences):

1. We select a *reference view*  $I_0 \in \mathcal{V}$  and an initial frame  $\Omega_0$  in  $I_0$ . Then, given an alternate view  $I_i \in \mathcal{V}$ , we map the points  $x$  of the frame  $\Omega_0$  to points  $y = h(x)$  of the alternate view. To this end we use the three-dimensional ground truth in order to estimate the actual *motion* of the pixels from (2.1), which does not depend on the local appearance. Note that  $h(x)$  is well defined even when some pixels  $y = h(x)$  are occluded.
2. We search for the warp  $w \in \mathcal{W}$  that best approximates  $h$ , for example by solving

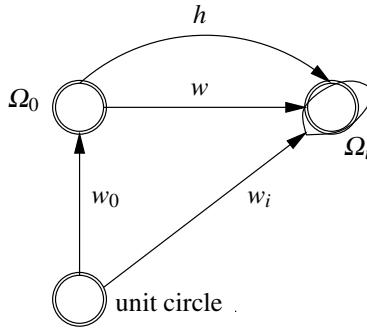
$$w = \operatorname{argmin}_{v \in \mathcal{W}} \int_{\Omega_0} \|h(x) - v(x)\|^2 dx. \quad (2.2)$$

Algorithms that solve efficiently this problem for the transformation classes  $\mathcal{W}$  of interest are reported in Appendix 2.5. Notice that one can choose a cost different from (2.2), and we illustrate a different example in (2.3).

3. We map the frame  $\Omega_0$  to the frame  $\Omega_i = w\Omega_0$  by the estimated warp  $w$ .

### Occlusions and Foldings

The procedure we have delineated is simple, but can be inadequate if the frame  $\Omega_0$  contains an occlusion or a strong depth discontinuity, which induces a highly non-linear or discontinuous motion  $h(x)$ . In such cases, instead of trying to capture the motion of all pixels simultaneously, one can expect the detector to track only the *dominant motion*, i.e., the motion of the background or the foreground, depending on which one occupies the larger portion of the region, or “patch.” To this end, before executing step (2.2) we consider splitting the patch in half. We sort the pixels  $x \in \Omega_0$  by depth and we search for a (relative) gap in the sorted values which is bigger than a threshold. If we find it, we restrict equation (2.2) only to the pixels before or after the split, based on majority rule.



**Fig. 2.2** Deviation. The figure illustrates the quality index (2.3). Intuitively, the deviation is the norm of the difference of the true motion  $h$  and the estimated motion  $w$ , normalized by projecting on the unit disk (canonical configuration). Normalization reflects the fact that features are being canonized before the descriptor is computed.

## Quality Indices

The procedure just delineated finds, in each image  $I_i$  of a view set, the best matching frame  $\Omega_i$ . However, not all matches are equal. Some may approximate very well the underlying image transformation, while others may be poor fits, due for instance to occlusions or strong non-linear distortions. For the evaluation of a real-world detector, it is important to assess which of these ground-truth matches are close to the idealized working assumptions, and which are not. To this end, we propose the following quality indices:

**Deviation.** This index measures the “non-linearity” of the warp. Let  $w_0 = (A_0, T_0)$  and  $w_i = (A_i, T_i)$  be the affine transformations that map the unit (oriented) circle on the reference frame  $\Omega_0$  and the alternate frame  $\Omega_i$ ; let  $w$  be the companion warp  $\Omega_i = w\Omega_0$  that approximates the true motion  $h(x)$ . The deviation index is a normalized version of the average square residual  $|h(x) - w(x)|^2$ , obtained by conjugation with  $w_i$ :

$$\text{dev}(w, h, \Omega_i) = \frac{1}{\pi} \int_{\{x:|x|<1\}} |w_i^{-1} \circ (h \circ w^{-1}) \circ w_i(x) - w_i^{-1} \circ w_i(x)|^2 dx. \quad (2.3)$$

The formula has a simple interpretation (Figure 2.2). It is the average squared residual  $|h(x) - w(x)|^2$  remapped to the canonized version of the frame  $\Omega_i$ . Noting that, by definition,  $w = w_i w_0^{-1}$  and all but  $h$  are affine warps, we find (see Appendix 2.5)

$$\text{dev}(w, h, \Omega_i) = \frac{1}{\pi} \int_{\{x:|x|<1\}} |A_i^{-1} (h \circ w_0(x) - w_i(x))|^2 dx. \quad (2.4)$$

In practice, we estimate the values of  $h$  on the pixels  $\hat{x}_i$  of the region  $\Omega_0$ ; in this case we use the formula

$$\text{dev}(w, h, \Omega_i) \approx \frac{1}{|\Omega_0|} \sum_{\tilde{x}_i \in \Omega_0} |A_i^{-1}(h(\tilde{x}_i) - w(\tilde{x}_i))|^2 \quad (2.5)$$

which preserves its validity even if the region  $\Omega_0$  intersects the image boundaries.

**Visibility.** This is the portion of the frame  $\Omega_i$  that falls inside the image boundaries.

**Occlusion.** This is the portion of the region  $\Omega_0$  that is occluded in the alternate view  $I_i$ . Occluded pixels  $x \in \Omega_0$  are determined empirically by checking whether their pre-image from the reference view  $I_0$  and the alternate view  $I_i$  correspond, i.e.,

$$R_{i0}\pi_0^{-1}(x) + T_{i0} \neq \pi_i^{-1}(h(x)).$$

**Splitting.** This is the portion of frame  $\Omega_0$  which is accounted for in the estimation of the dominant motion and ranges from 1 (complete frame) to 1/2 (half frame).

Figure 2.4 illustrates the quality indices for a number of co-variant frames.

### 2.3.4 Comparing Ground-Truth and Real-World Correspondences

One may regard a real-world detector as a mechanism that attempts to extract co-variant frames from the local appearance only. This task is difficult because, while the definition of correspondences (co-variant frames) is based on the knowledge of the ground-truth transformations  $h_{ij}$ , these are not available to the detector, and cannot be estimated by it as this would require operating on multiple images simultaneously [16].

There exist several mechanisms by which detectors are implemented in practice. The simplest one is to randomly extract a large number of feature frames so that eventually some frame sets will be filled “by chance”. Albeit very simple, this process poses a high computational load on the matching step. More refined approaches, such as Harris, SIFT, attempt to attach feature frames to specific patterns of the local image appearance (for example SIFT attaches oriented disks to “image blobs”). This enables the detector to explicitly “track” image transformations while avoiding the exhaustive approach of the random detectors. In general, constructing a co-variant detector requires that it be possible to associate co-variant frames to images based on the (local) appearance only. So, for example, we can associate disks to image “blobs,” as long as we make sure that, as the blobs move, the disks move according.

No matter what the underlying principle on which a detector is based, the quality of the correspondences established by a real-world detector can be expected to be much lower than the ideal correspondences introduced in Section 2.3.3, which, under the limited expressive power of the regions extracted (e.g., disks are limited to similarity transformations), optimally approximate the actual image

transformations. Thus ground-truth frame sets can be used to compare and evaluate the performance of the real-world detectors.

To assess the performance of a detector, we therefore measure how much the approximate co-variant frame  $\tilde{\Omega}_i$  extracted by the detector deviates from the ground truth co-variant frame  $\Omega_i$  defined in Section 2.3.3. To this end, we use the same criterion introduced in 2.3, and compare the deviation of the ground truth and estimated motions. Consider first the simpler case in which frames are complete for the companion transformations  $\mathcal{W}$  (for example oriented disks for similarity transformations). Let  $w_i = (A_i, T_i)$  and  $\tilde{w}_i = (\tilde{A}_i, \tilde{T}_i)$  be the unique (by hypothesis) warps in  $\mathcal{W}$  that bring the oriented unit circle to the frames  $\Omega_i$  and  $\tilde{\Omega}_i$ . Let  $w = \tilde{w}_i w_i^{-1}$  be the transformation mapping  $\Omega_i$  to  $\tilde{\Omega}_i$ ; the desired transformation  $h$  is the identity  $\mathbf{1}$  and by plugging back into eq. (2.3) (see Appendix 2.5) we obtain the *oriented matching deviation*

$$\text{dev}(w, \mathbf{1}, \tilde{\Omega}_i) = \frac{1}{4} \|\tilde{A}_i^{-1} A_i - \mathbf{1}\|_F^2 + |\tilde{A}_i^{-1} (T_i - \tilde{T}_i)|^2 \quad (2.6)$$

where  $\|\cdot\|_F$  is the Frobenius matrix norm.

In case the frames are not oriented,  $w_i$  and  $\tilde{w}_i$  are known only up to right rotations  $R$  and  $\tilde{R}$  and we have

$$\text{dev}(w, \mathbf{1}, \tilde{\Omega}_i) = \frac{1}{4} \|\tilde{R}^\top \tilde{A}_i^{-1} A_i R - \mathbf{1}\|_F^2 + |\tilde{A}_i^{-1} (T_i - \tilde{T}_i)|^2 \quad (2.7)$$

where we used the fact that the Euclidean norm  $|\cdot|$  is rotationally invariant. We obtain the *un-oriented matching deviation* by minimizing over  $R$  and  $\tilde{R}$  (see Appendix 2.5)

$$\text{dev}(w, \mathbf{1}, \tilde{\Omega}_i) = \frac{1}{4} (\|\tilde{A}_i^{-1} A_i\|_F^2 + 2(1 - \text{tr}[\Lambda])) + |\tilde{A}_i^{-1} (T_i - \tilde{T}_i)|^2 \quad (2.8)$$

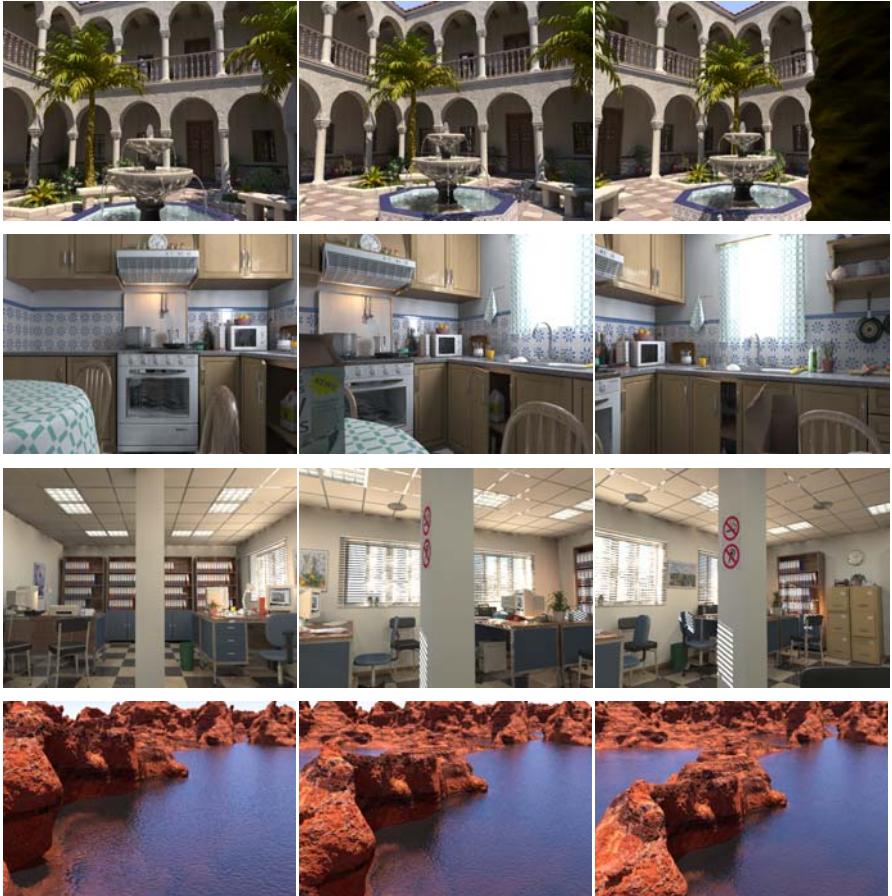
where  $\Lambda$  is the matrix of the singular values of  $\tilde{A}_i^{-1} \tilde{A}_i$ .

### 2.3.5 The Data

Based on the concepts that we have introduced in the previous sections, we now describe a new dataset to evaluate and learn visual invariants. The dataset is composed as follows:

#### View Sets

View sets are obtained from a number of three dimensional scenes shot from different vantage points (Figure 2.3). Each image comes with accurate geometric ground truth information in the form of a *depth map*. This data can be acquired by means of special instrumentation (e.g., a dome and a laser scanner), but in this work we propose to use high quality synthetic images instead. This has the advantages that (a) no special instrumentation is needed; (b) much more accurate ground truth can be generated; (c) automated data extraction procedures can be easily devised. Our

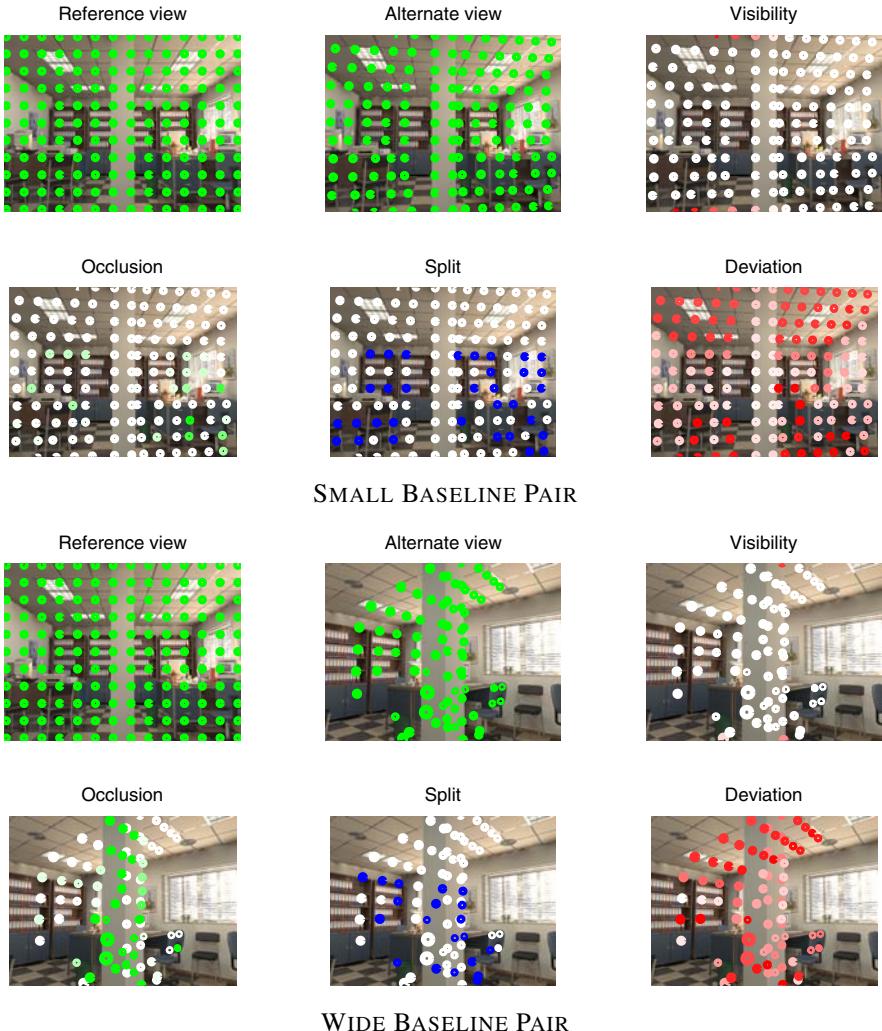


**Fig. 2.3** View sets. We show a small portion of a few view sets. These are synthetic rendering of scenes from [11] and come with accurate ground truth. Each image requires several CPU hours to be generated. The data set, which required a large computer cluster to be computed, is available to the public at [14].

data is largely based on publicly available 3-D scenes developed by [11] and generated by means of the freely available **POV-Ray** ray tracer.<sup>4</sup> Currently we work with a few such scenes that include natural as well as man-made environments; for each scene we compute a large number of views (from 300 to 1000) together with their depth map and camera calibration. The camera is moved to cover a large volume of space (it is more important to sample the camera translations rather than the camera rotations as additional orientations can be simulated exactly in post-processing by simple homographies).

---

<sup>4</sup> We actually use a customized version to export the required ground truth data. Patches are available from [14].

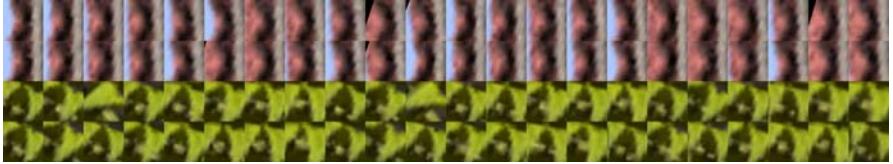


**Fig. 2.4** Quality indices. We show examples of the four quality indices (visibility, occlusion, split, and deviation) for a selection of features in a small baseline pair (top) and wide baseline pair (bottom). Quality indices signal if, and for what reason, a certain match deviates from the idealized working assumptions. Brighter colors indicate higher quality patches.

## Frame Sets

For each view set we compute a number of co-variant frame sets (Figure 2.5). This happens as follows:

- We choose a detector (e.g., SIFT) and a number of *reference views* in the view set.
- We run the detector on each reference view to extract reference frames.



**Fig. 2.5** Frame sets (correspondences). We show portion of two ground-truth frame sets (Section 2.3.1) as canonized patches. Each patch is obtained by un-warping to a canonical configuration the corresponding co-variant frame. Note that, due to complex reflectance and geometric phenomena, canonization never yields perfectly aligned patches.

- We re-map each reference frame to all other views as explained in Section 2.3.3 and we compute the four quality indices. The resulting collection of frames is a co-variant frame set. Based on the quality indices, frames can be filtered out in order to generate data of varying difficulty.
- Optionally, we run the detector on each non-reference view as well and we match each co-variant frame to a detected frame by minimizing the quality index introduced in Section 2.3.4. We then record the matching score and matched frames along with the co-variant frame set. This is the approximation of the co-variant frame set obtained from the real-world detector.

In practice, only a few reference views (from 2 to 4) are selected for each view set. This alone is sufficient to generate several thousand frame sets, and most frame sets count dozens of frames from different views. Eventually it is easy to generate data in the order of millions frames. The data comes with quality indices so that interesting subsets can be easily extracted.

## 2.4 Learning to Compare Invariant Features

The data developed in Section 2.3 can be used to:

1. Learn natural deformation statistics, similarly to [13], but in a wide-baseline setting.
2. Evaluate/learn detectors that compute good approximations of co-variant frames.
3. Evaluate/learn descriptors, given either the co-variant frame sets or the frame sets matched to the output of any practical co-variant detector.
4. Evaluate/learn similarity measures between descriptors.

Here we limit ourselves to the last task for the purpose of illustration of the use of the dataset. While the improvements we expect are limited, since we are only operating on the last ingredient of the feature matching pipeline, the results are readily applicable to existing systems.

More concretely, given a frame  $\Omega_0$  in a reference view  $I_0$  and an alternate view  $I_1$ , we study two problems: (i) how to find the frame  $\Omega_1$  of  $I_1$  that matches  $\Omega_0$  (Section 2.4.2) and (ii) when to accept a putative match  $\Omega_0 \leftrightarrow \Omega_1$  in order to minimize



**Fig. 2.6** Learning SIFT metric. We show four views of a co-variance frame (the frame on the ceiling lamp) and the ten most similar SIFT features in term of their SIFT descriptors. Shades of green are proportional to the descriptor  $\phi_2$ -similarity to the reference descriptor.

the expected risk of making a mistake (Section 2.4.3). We focus on SIFT features (both detector and descriptor) because of their popularity, but any other similar technique could be studied in this fashion.

### 2.4.1 Wide Baseline Motion Statistics

[13] studies the statistic of optical flow-based on simulated visual data. However, the analysis is limited to small baseline motion; our data is characterized by a much larger variety of viewing conditions, which enable us to collect statistic on wide-baseline motion.

Here we propose to study the residual of the pixel motion after canonization, i.e., after the companion transformation has been removed, as in (2.2):

$$\forall x : |x| < 1 : \quad r(x) \doteq w_i^{-1} \circ (h \circ w^{-1}) \circ w_i(x) - w_i^{-1} \circ w_i(x)$$

where  $w_i(x)$  maps the unit circle to the co-variance frame  $\Omega_0$ ,  $h(x)$  is the ground-truth motion and  $w(x)$  is the companion transformation.

### 2.4.2 Learning to Rank Matches

Given a frame  $\Omega_0$  of the reference view  $I_0$ , its descriptor  $f_0$  and an alternate view  $I_1$ , we order the frames  $\Omega_1, \Omega_2, \dots$  of  $I_1$  based on the similarity  $\phi$  of their descriptors  $f_1, f_2, \dots$  to  $f_0$ , i.e.,

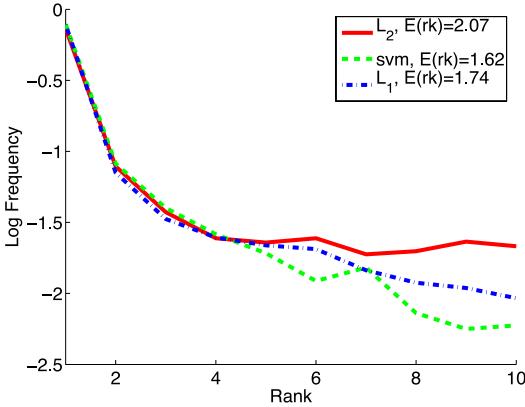
$$\phi(f_0, f_1) \leq \phi(f_0, f_2) \leq \dots$$

Ideally the similarity function  $\phi$  is chosen in such a way that the correct matching frame  $\Omega_i$  is ranked first.

Normally the similarity of a pair of SIFT descriptors is just their  $L_1$  or  $L_2$  distance, i.e.,

$$\phi_p(f_0, f_1) \doteq \|f_0 - f_1\|_p, \quad p = 1, 2.$$

Here we show how a similarity  $\phi$  can be learned that outperforms both  $\phi_1$  and  $\phi_2$ . We do this by setting up a learning problem as follows: Based on our ground-truth data, we sample pairs of corresponding descriptors  $(f_0, f_1)$  from a frame set and



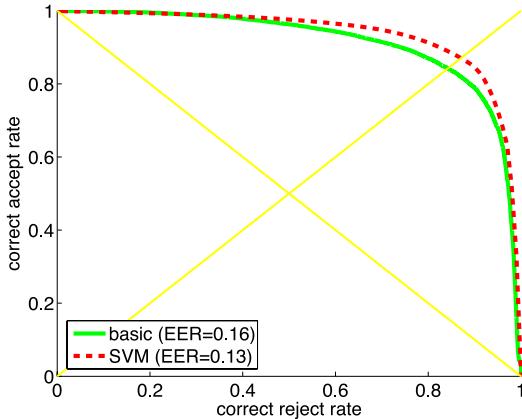
**Fig. 2.7** Learn to rank matches. The figure shows the distribution of the rank values of the correct feature match, averaged for frame sets in our database. The SVM-based ranking outperforms the naive  $\phi_1$  and  $\phi_2$  ranking, resulting in an expected rank of 1.62, 1.74 and 2.07 respectively.

a pair of non-corresponding descriptors  $(f_0, f)$  randomly. We then learn a binary classifier  $D(f_0, f_1)$  for the task of deciding whether  $f_0$  and  $f_1$  are the descriptors of corresponding features. Following [3], we assume that the classifier is in the form  $[\phi(f_0, f_1) \leq \tau]$  for some function  $\phi$  (for example this is the case for a support vector machine (SVM) but one could use boosting as well [18]) and we use  $\phi$  as a similarity measure.

**Re-ranking.** Since the goal is to improve  $\phi_1$  and  $\phi_2$  (which have already good performance), instead of choosing negative descriptors  $f$  completely at random, we select them among the descriptors of the alternate view that have  $\phi_p$ -rank smaller or equal to 10 (Figure 2.6). In testing, the learned similarity  $\phi$  is then used to re-rank these top matches in hope of further improving their ranking. This approach has several benefits: (a) since the computation of  $\phi$  is limited to a few features, testing speed is not a concern; (b) experimentally we verified that the top ten features include very often the correct match; (c) the learning problem has a much more restricted variability because features are  $\phi_p$ -similar by construction.

**Learning.** We select about 500 frame sets (matched to actual SIFT frames – see Section 2.3.4) and we extract their reference frames  $\Omega_0$  and descriptors  $f_0$ ; for each of them we select about 10 alternate views and we extract the relative co-variance frame  $\Omega_1$  and descriptor  $f_1$ . In this way, we form about 5,000 positive learning pairs  $(f_0, f_1)$ . For each positive pair  $(f_0, f_1)$ , we add about 10 negative pairs  $(f_0, f)$  formed as explained for a total of 55,000 examples. The data is used to learn an SVM with polynomial kernel.

**Testing.** While  $\phi$  is optimized for classification, here we are interested in its ranking performance. Thus testing is done by taking a large number of fresh frame sets and



**Fig. 2.8** Learning to accept matches. The figure compares the ROC curves of the function  $D(f_0, f_1, f_2)$  in its basic form and as learned by an SVM.

averaging the ranking performance of  $\phi$  over them. Figure 2.7 shows that learning can indeed improve the basic similarities.

### 2.4.3 Learning to Accept Matches

Once putative matches have been proposed, for example based on the similarity metric  $\phi$ , we need to accept or reject them based on some notion of expected risk. In some applications we also want to order matches by reliability [1]. [6] proposes a simple score that can be used to both accept and rank putative matches. With notation similar to the previous section, denote  $f_0$  the descriptor of a reference frame  $\Omega_0$  and by  $f_1$  and  $f_2$  the two  $\phi$ -top matches in an alternate view. We define the belief that the match  $\Omega_0 \leftrightarrow \Omega_1$  is correct as

$$P(\Omega_0 \leftrightarrow \Omega_1 | f_0, f_1, f_2) = 1 - \frac{\phi(f_1, f_0)}{\phi(f_2, f_0)}.$$

Here we use  $\phi = \phi_2$  to be compatible with [6]. This quantity can be directly used to rank and accept matches, the latter by comparing it to a threshold  $\tau$ , getting the decision function

$$D(f_0, f_1, f_2) = [P(\Omega_0 \leftrightarrow \Omega_1 | f_0, f_1, f_2) \leq \tau]. \quad (2.9)$$

As  $D(f_0, f_1, f_2)$  is a decision function, it can also be learned by means of some standard technique, which we do next.

**Data and learning.** Data is obtained similarly to Section 2.4.2 with the obvious adaptations. Learning is still performed by an SVM based on a polynomial kernel.

**Testing.** In Figure 2.8 we plot the ROC curve of (2.9) as  $\tau$  is varied and the ROC curve of the SVM-based decision function  $D(f_0, f_1, f_2)$ . The equal error rate is lowered from 0.16 to 0.13 showing again that learning can be used to improve the basic method.

## 2.5 Discussion

We have presented an extensive, flexible, accurate ground-truthed dataset for matching local invariant features. Together with it, we have presented a methodology to evaluate local features, and illustrated their use to not only evaluate, but also improve current algorithms. Our analysis separates the effects of a feature detector, a descriptor, and a matching algorithm, and our dataset is aimed at facilitating the collection of natural image deformation statistics induced by viewpoint changes, and at incorporating them in the design of better features. A similar procedure can be followed to incorporate natural reflectance, illumination and occlusion statistics, which is obviously beyond the scope of this Chapter. We have demonstrated the use of the dataset to improve on the matching score in matching SIFT features. Albeit the quantitative improvement is not stunning, it is sufficient to illustrate the potential advantage associated in the use of the dataset and the associated methodology for evaluating local features.

## Appendix 1: Calculations

### Justification of Equation (2.5)

By changing variable in (2.4) we obtain

$$\begin{aligned} \text{dev}(w, h, \Omega_i) &= \frac{1}{\pi \det A_0} \int_{\Omega_0} |A_i^{-1}(h(\tilde{x}) - w(\tilde{x}))|^2 d\tilde{x} \\ &\approx \frac{1}{\pi \det A_0} \sum_{\tilde{x}_i \in \Omega_0} |A_i^{-1}(h(\tilde{x}_i) - w(\tilde{x}_i))|^2. \end{aligned}$$

Note that  $\pi \det A_0$  is just the area of the region  $\Omega_0$ , so we obtain (2.5).

### Oriented Matching Deviation and Frobenius Norm

Define the “size” of the linear deformation  $A \in GL(2)$  the quantity

$$\|A\|^2 = \frac{1}{\pi} \int_{|x|<1} x^\top A^\top A x dx.$$

This is the average of the norm of the vector  $Ax$  as  $x$  is moved along the unit circle. We have

$$\|A\|^2 = \frac{1}{\pi} \text{tr} \left[ A^\top A \int_{|x|<1} x x^\top dx \right] = \frac{1}{4} \text{tr}[A^\top A]$$

so this is just the Frobenius norm of  $A$  (up to a scale factor). Now consider the affine deformation  $Ax + T$ . We define analogously

$$\begin{aligned} \pi\|(A, T)\|^2 &= \int_{|x|<1} |Ax + T|^2 dx \\ &= \int_{|x|<1} x^\top A^\top Ax dx + 2 \int_{|x|<1} T^\top Ax dx + \int_{|x|<1} T^\top T dx. \end{aligned}$$

So the “Frobenius norm” of an affine deformation is

$$\|(A, T)\|^2 = \frac{1}{\pi} \int_{|x|<1} |Ax + T|^2 dx = \frac{1}{4} \text{tr}[A^\top A] + |T|^2.$$

This also justifies (2.6) because

$$\begin{aligned} \text{dev}(w, \mathbf{1}, \tilde{\Omega}_i) &= \frac{1}{\pi} \int_{\{x:|x|<1\}} |\tilde{A}_i^{-1}(w_i(x) - \tilde{w}_i(x))|^2 dx \\ &= \frac{1}{\pi} \int_{\{x:|x|<1\}} |(\tilde{A}_i^{-1}A_i - \mathbf{1})x + \tilde{A}_i^{-1}(T_i - \tilde{T}_i)|^2 dx. \end{aligned}$$

## Unoriented Matching Deviation

**Lemma 2.1.** Let  $A$  be a square matrix and  $Q$  a rotation of the same dimension and let  $U\Lambda V^\top = A$  be the SVD of  $A$ . Then the rotation  $Q$  which minimizes the quantity  $\text{tr}[QA]$  is  $UV^\top$  and the minimum is  $\text{tr}[\Lambda]$ .

*Proof.* Let  $V\Lambda U^\top = A$  be the SVD decomposition of matrix  $A$ . We have  $\text{tr}[QA] = \text{tr}[Q\Lambda]$  where  $\Lambda$  is a diagonal matrix with non-negative entries and  $L = U^\top QV$  is a rotation matrix. The trace is equal to  $\sum_i L_{ii}\lambda_i$  where  $0 \leq L_{ii} \leq 1$  and  $L_{ii} = 1$  for all  $i$  if, and only if,  $L$  is the identity. So the optimal value of  $Q$  is  $Q = UV^\top$ .

Since the Frobenius norm is rotationally invariant, (2.7) can be written as

$$\|\tilde{R}^\top \tilde{A}_i^{-1} A_i R - \mathbf{1}\|_F^2 = \|\tilde{A}_i^{-1} \tilde{A}_i\|_F^2 - 2 \text{tr}[Q \tilde{A}_i^{-1} A_i] + 2, \quad Q = R \tilde{R}_i^\top.$$

Minimizing this expression with respect to  $Q$  is equivalent to maximizing the term  $\text{tr}[Q \tilde{A}_i^{-1} A_i R]$ . Let  $V\Lambda U^\top = \tilde{A}_i^{-1} \tilde{A}_i$  be the SVD of  $\tilde{A}_i^{-1} A_i$ ; Lemma 2.1 shows that the maximum is  $\text{tr}[\Lambda]$  (obtained for  $Q = UV^\top$ ), yielding (2.8).

## Appendix 2: Algorithms

In this Section we derive algorithms to minimize (2.2) in the cases of interest. The purpose of the following algorithm is to align a set of points  $x_1^{(1)}, \dots, x_1^{(K)}$  to a set of points  $x_2^{(1)}, \dots, x_2^{(K)}$  up to either an affine, rigid, or similarity motion.

## Alignment by an Affine Motion

Let  $x_2 = Ax_1 + T$  for an affine motion  $(A, T)$ . We can transform this equation as

$$x_2 = Ax_1 + T = Bx = (x^\top \otimes I_{2 \times 2}) \text{vec } B, \quad x = \begin{bmatrix} x_1 \\ 1 \end{bmatrix}, \quad B = [A \ T]$$

where  $\otimes$  is the Kronecker product and  $\text{vec}$  is the stacking operator. We obtain one of these equations for each of the points  $x_1^{(k)}$ ,  $k = 1, \dots, K$  to be aligned and solve them in the least-squares sense for the unknown  $B$ .

## Alignment by a Rigid Motion

We give first a closed-form sub-optimal algorithm. This algorithm is the equivalent as the one proposed in [19], but our development is straightforward.

Let  $x_2 = Rx_1 + T$  be a rigid motion  $(R, T)$  and assume for the moment that the points are three dimensional. Let  $R = \exp(\theta\hat{r})$  where  $r$ ,  $|r| = 1$  is the axis of rotation,  $\hat{r}$  is the hat operator [7], and  $\theta > 0$  is the rotation angle. We use Rodrigues' formula [7]  $R = I + \sin \theta \hat{r} + (1 - \cos \theta) \hat{r}^2$  to get

$$\begin{aligned} x_2 &= Rx_1 + T = x_1 + \sin \theta \hat{r}x_1 + (1 - \cos \theta)\hat{r}^2x_1 + T, \\ x_1 &= R^{-1}(x_2 - T) = x_2 - T - \sin \theta \hat{r}(x_2 - T) + (1 - \cos \theta)\hat{r}^2(x_2 - T). \end{aligned}$$

Adding the previous equations, collecting  $\sin \theta \hat{r}$ , and using the trigonometric identity  $\tan(\theta/2) = (1 - \cos \theta)/\sin \theta$  we obtain

$$\sin \theta \hat{r} \left( x_1 - x_2 + T + \tan \frac{\theta}{2} \hat{r}(x_1 + x_2 - T) \right) = 0.$$

It is easy to check that this condition is equivalent to  $x_2 = Rx_1 + T$  for  $|\theta| < \pi$ . A sufficient condition is

$$x_1 - x_2 + T + \tan \frac{\theta}{2} \hat{r}(x_1 + x_2 - T) = 0$$

which can be rewritten as

$$x_1 - x_2 + \tan \frac{\theta}{2} \hat{r}(x_1 + x_2) + z = 0, \quad z = T - \tan \frac{\theta}{2} \hat{r}T.$$

Since, no matter what  $r$  is,  $z$  spans all  $\mathbb{R}^3$  as  $T$  varies, we can equivalently solve this equation linear in the unknowns  $\tan(\theta/2)r$  and  $z$  in order to estimate the rigid transformation. As in the previous section, one obtains one of such equations for each of the points  $x_1^{(k)}$ ,  $k = 1, \dots, K$  to be aligned and finds the solution in the least-squares sense.

If there is noise in the model, i.e., if  $x_2 = Rx_1 + T + n$ , we get the condition

$$x_1 - x_2 + \tan \frac{\theta}{2} \hat{r}(x_1 + x_2) + z + \tan \frac{\theta}{2} \hat{r}n = -n.$$

This means that for moderate rotations (away from  $\pm\pi$ ) minimizing the  $l^2$  residual of this equation is almost equivalent to minimizing the norm of  $n$  itself. However if  $\theta$  approaches  $\pm\pi$ , then the term  $\tan(\theta/2)\hat{r}n$  will dominate, biasing the estimate.

The formulas work for the 2-D case with little adaptation. In this case we assume that all the points lie on the X-Y plane and the rotation vector is aligned to the Z axis, obtaining

$$x_1 - x_2 - \tan \frac{\theta}{2} \begin{bmatrix} x_{2,1} + x_{2,2} \\ -x_{1,1} - x_{1,2} \end{bmatrix} + z = 0.$$

Finally, the estimate can be refined by the iterative algorithm given in the next section (where one fixes the scale  $s$  to the constant 1).

### **Alignment by a Similarity**

There is no closed-form algorithm for this case. Instead, we estimate iteratively the translation  $T$  and the linear mapping  $sR$ . While the solution to the first problem is obvious, for the second consider the following equation:

$$\begin{aligned} \min_{s,R} \sum_k (x_2^{(k)} - sRx_1^{(k)})^\top (x_2^{(k)} - sRx_1^{(k)}) \\ = \min_{s,R} \sum_k |x_2^{(k)}|^2 - 2s \sum_k x_2^{(k)^\top} Rx_1^{(k)} + s^2 \sum_k |x_1^{(k)}|^2. \end{aligned} \quad (2.10)$$

Rewrite the cost function al as  $c - 2bs + as^2$ . The optimal value for  $s$  given a certain  $R$  is  $s^* = b/a$  and the optimal value of the cost function is  $a + c - 2b^2/a$ . Note that only the term  $b$  is a function of  $R$ , while neither  $a$  nor  $c$  depend on it. As a consequence, the optimal value of  $R$  is obtained by solving the problem

$$\max_R b = \max_R \sum_k x_2^{(k)^\top} Rx_1^{(k)} = \max_R \sum_k \text{tr} \left( Rx_1^{(k)^\top} x_2^{(k)} \right).$$

Thus we are simply maximizing the correlation of the rotated point  $Rx_1^{(k)}$  and the target points  $x_2^{(k)}$ . By taking the derivative of the trace w.r.t. the rotation angle  $\theta$ , we immediately find that the optimal angle is  $\theta^* = \text{atan}(w_2/w_1)$  where

$$w_1 = \sum_k |x_2^{(k)}| |x_1^{(k)}| \cos \theta^{(k)}, \quad w_2 = \sum_k |x_2^{(k)}| |x_1^{(k)}| \sin \theta^{(k)}$$

where  $\theta^{(k)}$  is the angle from vector  $x_1^{(k)}$  to vector  $x_2^{(k)}$ .

Thus, in order to estimate  $R$  and  $s$ , we can first solve for the optimal rotation  $R^*$ , and then solve for the scale, which is obtained as

$$s^* = \frac{b}{a} = \frac{\sum_k x_2^{(k),\top} R^* x_1^{(k)}}{\sum_k |x_1^{(k)}|^2}.$$

The convergence of the alternating optimization can be greatly improved by removing the mean from  $x_1^{(k)}$ ,  $k = 1, \dots, K$  as a pre-processing step.

## References

1. Chum, O., Matas, J.: Matching with PROSAC – progressive sample consensus. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2005)
2. Fraundorfer, F., Bischof, H.: A novel performance evaluation method of local detectors on non-planar scenes. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2005)
3. Hertz, T., Bar-Hillel, A., Weinshall, D.: Learning distance functions for image retrieval. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2004)
4. Lepetit, V., Lagger, P., Fua, P.: Randomized trees for real-time keypoint recognition. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2005)
5. Ling, H., Jacobs, D.W.: Deformation invariant image matching. In: Proceedings of the International Conference on Computer Vision (2005)
6. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 2(60), 91–110 (2004)
7. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.S.: An Invitation to 3-D Vision. Springer, Heidelberg (2003b)
8. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: Proceedings of the British Machine Vision Conference (2002)
9. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Van Gool, L.: A comparison of affine region detectors. International Journal of Computer Vision 1(60), 63–86 (2004)
10. Moreels, P., Perona, P.: Evaluation of features detectors and descriptors based on 3D objects. In: Proceedings of the International Conference on Computer Vision (2005)
11. Piquer, J.V.: The persistence of ignorance (2006), <http://www.ignorancia.org/>
12. Rockett, P.I.: Performance assesment of feature detection algorithms: A methodology and case study on corner detectors. Transaction on Image Processing 12(12) (2003)
13. Roth, S., Black, M.J.: On the spatial statistics of optical flow. In: Proceedings of the International Conference on Computer Vision (2005)
14. Vedaldi, A.: A ground-truthed dataset for evaluation and learning of viewpoint invariant features (2008), <http://vision.ucla.edu/gtpvi>
15. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms (2008), <http://www.vlfeat.org/>

16. Vedaldi, A., Soatto, S.: Features for recognition: Viewpoint invariance for non-planar scenes. In: Proceedings of the International Conference on Computer Vision (2005)
17. Winder, S.A.J., Brown, M.: Learning local image descriptors. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
18. Zhou, S.K., Shao, J., Georgescu, B., Comaniciu, D.: BoostMotion: Boosting a discriminative similarity function for motion estimation. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2006)
19. Zhuang, H., Sudhakar, R.: Simultaneous rotation and translation fitting of two 3-D point sets. Transaction on Systems, Man, and Cybernetics 27(1) (1997)

# Chapter 3

## Dynamic Graph Cuts and Their Applications in Computer Vision

Pushmeet Kohli and Philip H.S. Torr

**Abstract.** Over the last few years energy minimization has emerged as an indispensable tool in computer vision. The primary reason for this rising popularity has been the successes of efficient graph cut based minimization algorithms in solving many low level vision problems such as image segmentation, object reconstruction, image restoration and disparity estimation. The scale and form of computer vision problems introduce many challenges in energy minimization. In this chapter we address the problem of efficient and exact minimization of groups of similar functions which are known to be solvable in polynomial time. We will present a novel dynamic algorithm for minimizing such functions. This algorithm reuses computation from previous problem instances to solve new instances resulting in a substantial improvement in the running time. We will present the results of using this approach on the problems of interactive image segmentation, image segmentation in video, human pose estimation and segmentation, and measuring uncertainty of solutions obtained by minimizing energy functions.

### 3.1 Introduction

Many problems in computer vision and scene understanding can be formulated in terms of finding the most probable values of certain hidden or unobserved variables. These variables encode a desired property of the scene and can be continuous or discrete. For the case of discrete variables, these problems are commonly referred to as *labelling problems* as they involve assigning a label to the hidden variables.

---

Pushmeet Kohli

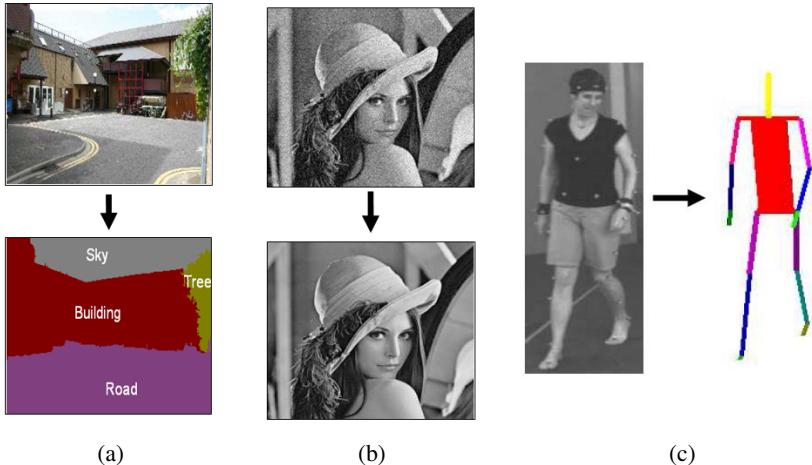
Microsoft Research, Cambridge, UK

e-mail: [p.kohli@microsoft.com](mailto:p.kohli@microsoft.com)

Philip H.S. Torr

Oxford Brookes University, Oxford, UK

e-mail: [philip.torr@brookes.ac.uk](mailto:philip.torr@brookes.ac.uk)



**Fig. 3.1** Some labelling problems in computer vision. (a) Object segmentation and recognition: Given any image, we want to find out which object each pixel in the image belongs to. There is one discrete random variable for each pixel in the image which can take any value from a set  $\mathcal{L}$  of object labels. For instance, we can use the object set {road, building, tree, sky}. (b) Image denoising: Given a noisy image of the scene, we want to infer the true colour of each pixel in the image. The problem is formulated in a manner similar to object segmentation. Again we use one discrete random variable per pixel which can take any value in RGB space. (c) Human pose estimation: Given an image, we want to infer the pose of the human visible in it. The problem is formulated using a vector of continuous pose variables which encode the orientation and different joint angles of the human.

Labelling problems occur in many forms, from lattice based problems of dense stereo and image segmentation [9, 66] to the use of pictorial structures for object recognition [18]. Some examples of problems which can be formulated in this manner are shown in Figure 3.1.

One of the major advances in computer vision in the past few years has been the use of efficient deterministic algorithms for solving discrete labelling problems. In particular, efficient graph cut based minimization algorithms have been extremely successful in solving many low level vision problems. These methods work by inferring the maximum a posteriori (MAP) solutions of conditional and markov random fields which are generally used to model these problems.

### 3.1.1 *Markov and Conditional Random Fields*

Random fields provide an elegant probabilistic framework to formulate labelling problems. They are able to model complex interactions between hidden variables in a simple and precise manner. The power of this representation lies in the fact that the probability distribution over different labellings of the random variables factorizes, and thus allows efficient inference.

Consider a discrete random field  $\mathbf{X}$  defined over a lattice  $\mathcal{V} = \{1, 2, \dots, n\}$  with a neighbourhood system  $\mathcal{N}$ . Each random variable  $X_i \in \mathbf{X}$  is associated with a lattice point  $i \in \mathcal{V}$  and takes a value from the label set  $\mathcal{L} = \{l_1, l_2, \dots, l_k\}$ . The neighbourhood system  $\mathcal{N}$  of the random field is defined by the sets  $\mathcal{N}_i, \forall i \in \mathcal{V}$ , where  $\mathcal{N}_i$  denotes the set of all neighbours of the variable  $X_i$ . A clique  $c$  is a set of random variables  $\mathbf{X}_c$  which are conditionally dependent on each other. Any possible assignment of labels to the random variables is called a *labelling* or *configuration*. It is denoted by the vector  $\mathbf{x}$ , and takes values from the set  $\mathbf{L} = \mathcal{L}^n$ .

A random field is said to be a Markov random field (MRF) with respect to a neighbourhood system  $\mathcal{N} = \{\mathcal{N}_v | v \in \mathcal{V}\}$  if and only if it satisfies the positivity property:  $p(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in \mathcal{X}^n$ , and the Markovian property:

$$p(x_v | \{x_u : u \in \mathcal{V} - \{v\}\}) = p(x_v | \{x_u : u \in \mathcal{N}_v\}) \quad \forall v \in \mathcal{V}. \quad (3.1)$$

Here we refer to  $p(X = \mathbf{x})$  by  $p(\mathbf{x})$  and  $p(X_i = x_i)$  by  $p(x_i)$ . The pairwise MRF commonly used to model image labelling problems is shown in Figure 3.8.

A conditional random field (CRF) may be viewed as an MRF globally conditioned on the data. The conditional distribution  $p(\mathbf{x}|\mathbf{D})$  over the labellings of the CRF is a *Gibbs* distribution and can be written in the form:

$$p(\mathbf{x}|\mathbf{D}) = \frac{1}{Z} \exp\left(-\sum_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c)\right), \quad (3.2)$$

where  $Z$  is a normalizing constant known as the partition function, and  $\mathcal{C}$  is the set of all cliques [42]. The term  $\psi_c(\mathbf{x}_c)$  is known as the potential function of the clique  $c$  where  $\mathbf{x}_c = \{x_i, i \in c\}$  is the vector encoding the labelling of the variables constituting the clique. The corresponding Gibbs energy is given by

$$E(\mathbf{x}) = -\log p(\mathbf{x}|\mathbf{D}) - \log Z = \sum_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c) \quad (3.3)$$

The most probable or maximum a posteriori (MAP) labelling  $\mathbf{x}^*$  of the random field is defined as

$$\mathbf{x}^* = \arg \max_{\mathbf{x} \in \mathbf{L}} p(\mathbf{x}|\mathbf{D}). \quad (3.4)$$

and can be found by minimizing the energy function  $E$ . This equivalence to MAP inference has made discrete energy minimization extremely important for problems which are solved using probabilistic methods.

Minimizing a discrete function is one of the core problems of optimization. Many combinatorial problems such as MAXCUT and constraint satisfaction (CSP) can be formulated in this manner. Although minimizing a function is NP-hard in general, there exist families of energy functions for which this could be done in polynomial time. Submodular set functions constitute one such well studied family. The algorithms for minimizing general functions belonging to this class of functions have high runtime complexity. This characteristic renders them useless for most computer vision problems which involve large number of random variables. Functions belonging to certain subclasses of submodular functions can be solved relatively

easily (i.e., are less computationally expensive to minimize). For instance, certain families of functions can be minimized by solving a st-mincut problem for which fast and efficient algorithms are available [8, 19, 27, 56].

## 3.2 Graph Cuts for Energy Minimization

Graph cuts have been extensively used in computer vision to compute the maximum a posteriori (MAP) solutions for various discrete pixel labelling problems such as image restoration, segmentation, voxel occupancy and stereo [7, 10, 28, 29, 37, 53, 75]. Greig *et al.* [24] were one of the first to use graph cuts in computer vision. They showed that if the pairwise potentials of a two label *pairwise* MRF were defined as an Ising model, then its exact MAP solution can be obtained in polynomial time by solving a st-mincut problem.

One of the primary reasons behind the growing popularity of graph cuts is the availability of efficient algorithms for computing the maximum flow (max-flow) in graphs of arbitrary topology [2, 8]. These algorithms have low polynomial runtime complexity, and enable fast computation of the minimum cost st-cut (st-mincut) problem. This in turn allows for the computation of globally optimal solutions for important classes of energy functions which are encountered in many vision problems [38, 33]. Even in problems where they do not guarantee globally optimal solutions, these algorithms can be used to find solutions which are strong local minima of the energy [9, 39, 32, 69]. These solutions for certain problems have been shown to be better than the ones obtained using other inference methods [66].

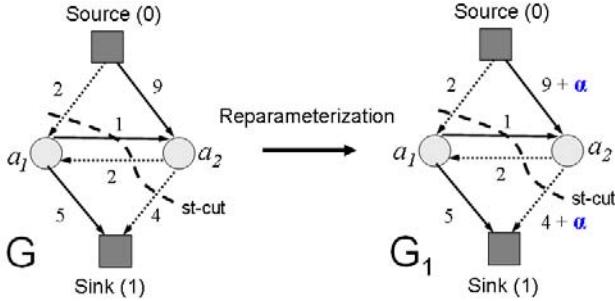
### 3.2.1 The st-Mincut Problem

In this Section we provide a general overview of the st-mincut/maxflow problem and give the graph notation used in this Chapter. A directed weighted graph  $G(V, E, C)$  with non-negative edge weights, is defined by a set of nodes  $V$ , a set of directed edges  $E$ , and an edge cost function  $C : E \rightarrow \mathbb{R}$  which maps each edge  $(i, j)$  of the graph to a real number  $c_{ij}$ <sup>1</sup>. We will use  $n$  and  $m$  to denote the number of nodes  $|V|$  and the number of edges  $|E|$  in the graph respectively. Graphs used in the st-mincut problem have certain special nodes called the terminal nodes, namely the source  $s$ , and the sink  $t$ . The edges in the graph can be divided into two disjoint categories: t-edges which connect a node to a terminal node, and n-edges which connect nodes other than the terminal nodes with each other. We make the following assumptions in our notation:  $(i, j) \in E \Rightarrow (j, i) \notin E$ , and  $(s, i) \in E \wedge (i, t) \in E$  for all  $i \in V$ . These assumptions are non-restrictive as edges with zero edge weights are allowed in our formulation. Thus we can conform to our notation without changing the problem.

A *cut* is a partition of the node set  $V$  into two parts  $S$  and  $\bar{S} = V - S$ , and is defined by the set of edges  $(i, j)$  such that  $i \in S$  and  $j \in \bar{S}$ . The cost of the cut  $(S, \bar{S})$  is given as:

---

<sup>1</sup> We will restrict our attention to edge cost functions of the form  $C : E \rightarrow \mathbb{R}^+ \cup \{0\}$ .



**Fig. 3.2** Graph reparameterization. The figure shows a graph  $G$ , and its reparameterization  $G_1$  obtained by adding a constant  $\alpha$  to both the t-edges of node  $a_2$ . The edges included in the st-mincut are depicted by dotted lines. Observe that although the cost of the st-mincut in  $G$  and  $G_1$  is different, the st-mincut includes the same edges for both graphs and thus induces the same partitioning of the graph.

$$C_{S,\bar{S}} = \sum_{i \in S, j \in \bar{S}} c_{ij}. \quad (3.5)$$

An st-cut is a cut satisfying the properties  $s \in S$  and  $t \in \bar{S}$ . Given a directed weighted graph  $G$ , the st-mincut problem is that of finding a st-cut with the smallest cost. By the Ford-Fulkerson theorem [20], this is equivalent to computing the maximum flow from the source to the sink with the capacity of each edge equal to  $c_{ij}$  [2].

### 3.2.2 Formulating the Max-Flow Problem

For a network  $G(V,E)$  with a non-negative capacity  $c_{ij}$  associated with each edge, the max-flow problem is to find the maximum flow  $f$  from the source node  $s$  to the sink node  $t$  subject to the edge capacity (3.6) and mass balance (3.7) constraints:

$$0 \leq f_{ij} \leq c_{ij} \quad \forall (i,j) \in E, \text{ and} \quad (3.6)$$

$$\sum_{i \in N(j)} (f_{ji} - f_{ij}) = 0 \quad \forall j \in V \quad (3.7)$$

where  $f_{ij}$  is the flow from node  $i$  to node  $j$  and  $N(j)$  is the neighbourhood of node  $j$  i.e.,  $N(j)$  consists of all nodes connected by an edge to  $j$  [2].

Observe that we can initialize the flows in the t-edges of any node  $i$  of the graph as  $f_{si} = f_{it} = \min(c_{si}, c_{it})$ . This corresponds to pushing flow through these edges from the source to the sink and has no effect on the final solution of the st-mincut problem. From this it can be deduced that the solution of the st-mincut problem is invariant to the absolute value of the terminal edge capacities  $c_{si}$  and  $c_{it}$ . It only depends on the difference of these capacities ( $c_{it} - c_{si}$ ). Adding or subtracting a constant to these capacities changes the objective function by a constant and does not affect the

overall st-mincut solution as can be seen in Figure 3.2. Such transformations result in a reparameterization of the graph and will be explained later in Section 3.3.

### 3.2.3 Augmenting Paths, Residual Graphs

Given a flow  $f_{ij}$ , the residual capacity  $r_{ij}$  of an edge  $(i, j) \in E$  is the maximum additional flow that can be sent from node  $i$  to node  $j$  using the edges  $(i, j)$  and  $(j, i)$  or formally  $r_{ij} = c_{ij} - f_{ij} + f_{ji}$ . A residual graph  $G(f)$  of a weighted graph  $G$  consists of the node set  $V$  and the edges with positive residual capacity (with respect to the flow  $f$ ). An augmenting path is a path from the source to the sink along unsaturated edges of the residual graph.

### 3.2.4 Minimizing Functions Using Graph Cuts

The problem of finding the minimum cost st-cut (st-mincut) in any graph can be written in terms of minimizing a sum of functions defined on individual and pairs of binary variables. Conversely, any submodular function of binary or *boolean* variables which can be written as a sum of unary and pairwise terms can be minimized by finding the st-mincut in the corresponding graph. In this Chapter, we will call functions of this form ‘second order functions’ or ‘functions of order 2’.

**Definition 3.1.** We say that a function  $f : \mathcal{L}^n \rightarrow \mathbb{R}$  is of order  $k$  if it can be written in terms of a sum of functions  $f_i : \mathcal{L}^k \rightarrow \mathbb{R}$ , each of which is defined on at most  $k$  variables.

In the above definition we use the function representation which leads to the smallest order.

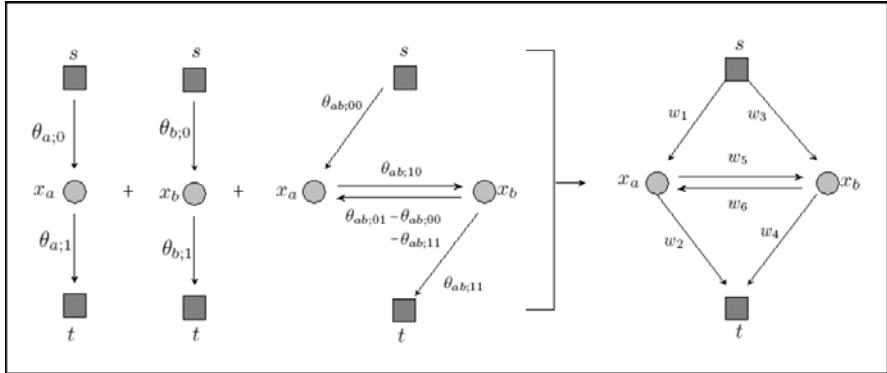
*Example 3.1.* The function  $f^a(x_1, x_2, x_3) = 4x_1 + 5x_2\bar{x}_3 + 3\bar{x}_2$  is of order 2 because of the maximal order term  $5x_2\bar{x}_3$ . Similarly, the function

$$f^a(x_1, x_2, x_3) = 4x_1 + 5\bar{x}_1x_2\bar{x}_3 \quad (3.8)$$

is of order 3 because of the maximal term  $5\bar{x}_1x_2\bar{x}_3$ .

Algorithms for finding the st-mincut require that all edges in the graph have non-negative weights. This condition results in a restriction on the class of energy functions that can be solved in this manner. For instance, binary second order functions can be minimized by solving a st-mincut problem only if they are submodular [38].

We will now show how second order functions of binary variables (also referred as Pseudo boolean functions) can be minimized by solving a st-mincut problem. The procedure for energy minimization using graph cuts comprises of building a graph in which each st-cut defines a configuration  $\mathbf{x}$ . The cost of an st-cut is equal to the energy  $E(\mathbf{x}|\theta)$  of its corresponding configuration  $\mathbf{x}$ . Finding the minimum cost st-cut in this graph thus provides us with the configuration having the least energy. Kolmogorov and Zabih [38] described the procedure to construct graphs for



**Fig. 3.3** Energy minimization using graph cuts. The Figure shows how individual unary and pairwise terms of an energy function taking two binary variables are represented and combined in the graph. Multiple edges between the same nodes are merged into a single edge by adding their weights. For instance, the cost  $w_1$  of the edge  $(s, x_a)$  in the final graph is equal to:  $w_1 = \theta_{a;0} + \theta_{ab;00}$ . The cost of a st-cut in the final graph is equal to the energy  $E(\mathbf{x})$  of the configuration  $\mathbf{x}$  the cut induces. The minimum cost st-cut induces the least energy configuration  $\mathbf{x}$  for the energy function.

minimizing pseudo-boolean functions of order at most 3. The graph constructions for functions of multi-valued variables were given later by [27] and [56].

We now explain the graph construction for minimizing energies involving binary random variables. We use the notation of [35] and write a second order function as:

$$E(\mathbf{x}|\theta) = \theta_{\text{const}} + \sum_{v \in V, i \in \mathcal{L}} \theta_{v;i} \delta_i(x_v) + \sum_{(u,v) \in E, (j,k) \in \mathcal{L}^2} \theta_{uv,jk} \delta_j(x_u) \delta_k(x_v), \quad (3.9)$$

where  $\theta_{v;i}$  is the penalty for assigning label  $i$  to latent variable  $x_v$ ,  $\theta_{uv;ij}$  is the penalty for assigning labels  $i$  and  $j$  to the latent variables  $x_u$  and  $x_v$  respectively. Further, each  $\delta_j(x_v)$  is an indicator function, which is defined as:

$$\delta_j(x_v) = \begin{cases} 1 & \text{if } x_v = j, \\ 0 & \text{otherwise.} \end{cases}$$

Functions of binary variables (pseudo-boolean functions) can be written as:

$$\begin{aligned} E(\mathbf{x}|\theta) = \theta_{\text{const}} &+ \sum_{v \in V} (\theta_{v;1}x_v + \theta_{v;0}\bar{x}_v) \\ &+ \sum_{(u,v) \in E} (\theta_{st;11}x_u x_v + \theta_{st;01}\bar{x}_u x_v + \theta_{st;10}x_u \bar{x}_v + \theta_{st;00}\bar{x}_u \bar{x}_v). \end{aligned} \quad (3.10)$$

The individual unary and pairwise terms of the energy function are represented by weighted edges in the graph. Multiple edges between the same nodes are merged into a single edge by adding their weights. The graph construction for a two variable

energy function is shown in Figure 3.3. The constant term  $\theta_{\text{const}}$  of the energy does not depend on  $\mathbf{x}$  and thus is not considered in the energy minimization procedure. The st-mincut in this graph provides us with the minimum solution  $\mathbf{x}^*$ . The cost of this cut corresponds to the energy of the solution  $E(\mathbf{x}^*|\theta)$ . The labelling of a latent variable depends on the terminal it is disconnected from by the minimum cut. In our notation, if the node is disconnected from the source, we assign it the label zero and one otherwise.

### 3.3 Minimizing Dynamic Energy Functions Using Dynamic Graph Cuts

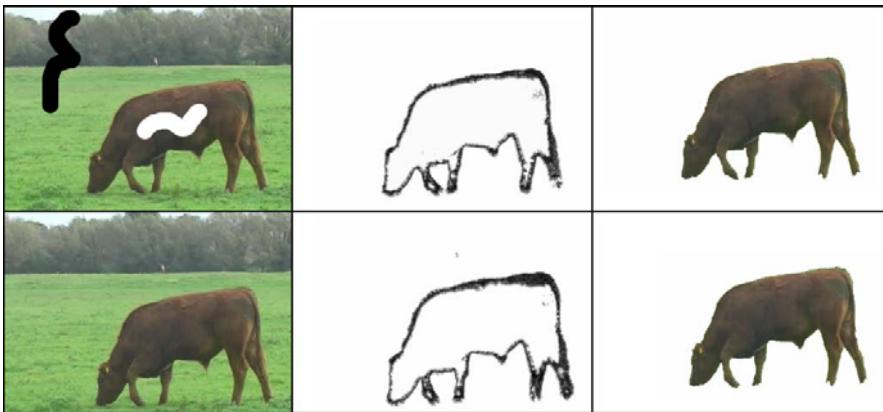
In many real world applications, multiple similar instances of a problem need to be solved sequentially (e.g., performing image segmentation on the frames of a video). The data (image) in this problem changes from one time instance to the next. Similarly, successive sub-problems solved in move making algorithms such as swap, expansion [9], or fusion move [44,74] are also similar.

Given the solution to an instance of the problem, the question arises as to whether this solution can help in solving other similar instances. In this Section we answer this particular question positively for functions that can be minimized exactly using graph cuts. Specifically, we show how the maxflow solution corresponding to an energy minimization problem can be used for efficiently minimizing another *similar* function with slightly different energy terms.

Our algorithm records the flow obtained during the computation of the max-flow corresponding to a particular problem instance. This recorded flow is used as an initialization in the process of finding the max-flow solution corresponding to the new problem instance (as seen in Figure 3.4). Our method belongs to a broad category of algorithms which are referred to as *dynamic*. These algorithms solve a problem by dynamically updating the solution of the previous problem instance. Their goal is to be more efficient than a recomputation of the solution after every change from scratch. Given a directed weighted graph, a *fully dynamic* algorithm should allow for unrestricted modification of the graph. The modification may involve addition and deletion of nodes and edges in the graph as well as changes in the cost (capacity) of any graph edge.

#### 3.3.1 Dynamic Computation

Dynamic algorithms are not new to computer vision. They have been extensively used in computational geometry for problems such as range searching, point location, convex hull, proximity and many others. For more on dynamic algorithms used in computational geometry, the reader is referred to [11]. A number of algorithms have been proposed for the dynamic mincut problem. Thorup [67] proposed a method which had a  $O(\sqrt{m})$  update time and took  $O(\log n)$  time per edge to list the cut edges. Here  $n$  and  $m$  denote the number of nodes and edges in the graph respectively. However, the dynamic *st-mincut* problem has remained relatively ignored.



**Fig. 3.4** Dynamic image segmentation using graph cuts. The images in the first column are two consecutive frames of the grazing cow video sequence. Their respective segmentations are shown in the third column. The first image in the first column also shows the user segmentation seeds (pixels marked by black (background) and white (foreground) colours). The user marked image pixels are used to learn histograms modelling foreground and background appearance (as in [7]). These histograms are used to compute a likelihood for each pixel belonging to a particular label. This likelihood is incorporated in the CRF used for formulating the image segmentation problem. The optimal segmentation solution (shown in column 3) is obtained by minimizing the energy function corresponding to the CRF. In column 2, we observe the n-edge flows obtained while minimizing the energy functions using graph cuts. It can be clearly seen that the flows corresponding to the two segmentations are similar. The flows from the first segmentation were used as an initialization for the max-flow problem corresponding to the second frame. The time taken for this procedure was much less than that taken for finding the flows from scratch.

Gallo *et al.* [22] introduced the problem of parametric max-flow and used a partially dynamic graph cut algorithm for the problem. Their algorithm had a low polynomial time complexity but was unable to handle arbitrary changes in the graph. Recently, Cohen and Tamassia [12] proposed a dynamic algorithm for the problem by showing how dynamic expression trees can be used for maintaining st-mincuts with  $O(\log m)$  time for update operations. However, their algorithm could only handle series-parallel digraphs.<sup>2</sup>

Boykov and Jolly [7] were the first to use a *partially* dynamic st-mincut algorithm in a vision application by proposing a technique with which they could update capacities of *certain* graph edges, and recompute the st-mincut dynamically. They used this method for performing interactive image segmentation, where the user could improve segmentation results by giving additional segmentation cues (seeds) in an online fashion. Specifically, they described a method for updating the cost of t-edges in the graph. In this Section we present a new fully dynamic algorithm

<sup>2</sup> Series-Parallel digraphs are graphs which are planar, acyclic and connected.

for the st-mincut problem which allows for arbitrary changes in the graph<sup>3</sup>. Juan and Boykov [30] proposed an algorithm in which instead of *reusing flow*, they used the st-mincut solution corresponding to the previous problem for solving a new st-mincut problem. Our work has been extended to the minimization of multi-label non-submodular function in [2].

An outline of the Section follows. The relationship between energy and graph reparameterization is explained in Section 3.3.2. Section 3.4 shows how exact st-mincut solutions of dynamically changing graphs can be efficiently computed by reusing flow. Specifically, it describes how the residual graph can be transformed to reflect the changes in the original graph using graph reparameterization, and discusses issues related to the computational complexity of the algorithm. In Section 3.5, we describe how the process of recomputing the st-mincut/max-flow can be further optimized by using *recycled* search trees.

### 3.3.2 Energy and Graph Reparameterization

We will now explain the concept of graph reparameterization which will be used later to show how we can minimize dynamic energy functions.

Recall from equation 3.10 that a second order energy function can be written in terms of an energy parameter vector  $\theta$  as:

$$\begin{aligned} E(\mathbf{x}|\theta) = & \theta_{\text{const}} + \sum_{v \in V} (\theta_{v;1}x_v + \theta_{v;1}\bar{x}_v) \\ & + \sum_{(u,v) \in E} (\theta_{st;11}x_u x_v + \theta_{st;01}\bar{x}_u x_v + \theta_{st;10}x_u \bar{x}_v + \theta_{st;00}\bar{x}_u \bar{x}_v). \end{aligned} \quad (3.11)$$

Two energy parameter vectors  $\theta_1$  and  $\theta_2$  are called reparameterizations of each other if and only if  $\forall \mathbf{x}, E(\mathbf{x}|\theta_1) = E(\mathbf{x}|\theta_2)$  [6,35,56,72]. This definition simply means that all possible labellings  $\mathbf{x}$  have the same energy under both parameter vectors  $\theta_1$  and  $\theta_2$ , and does not imply that  $\theta_1 = \theta_2$ . There are a number of transformations which can be applied to an energy parameter vector  $\theta$  to obtain its reparameterization  $\bar{\theta}$ . For instance the transformations given as:

$$\forall i \quad \bar{\theta}_{v;i} = \theta_{v;i} + \alpha, \quad \bar{\theta}_{\text{const}} = \theta_{\text{const}} - \alpha \quad \text{and} \quad (3.12)$$

$$\forall i, j \quad \bar{\theta}_{st;ij} = \theta_{st;ij} + \alpha, \quad \bar{\theta}_{\text{const}} = \theta_{\text{const}} - \alpha \quad (3.13)$$

result in the reparameterization of the energy parameter vector.

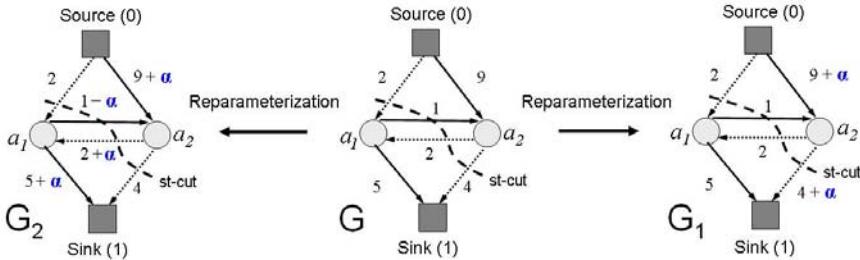
As both parameters  $\theta$  and  $\bar{\theta}$  define the same energy function, the minimum energy labelling for both will be the same i.e.

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} E(\mathbf{x}|\theta_1) = \arg \min_{\mathbf{x}} E(\mathbf{x}|\theta_2) \quad (3.14)$$

This in turn implies that the graphs constructed for minimizing the energy functions  $E(\mathbf{x}|\theta_1)$  and  $E(\mathbf{x}|\theta_2)$  (using the procedure explained in the previous Section)

---

<sup>3</sup> An earlier version of this Section appeared as [33].



**Fig. 3.5** Graph reparameterization. The Figure shows a graph  $G$ , its two reparameterizations  $G_1$  and  $G_2$  along with their respective st-mincuts. The edges included in the st-mincut are marked by dotted lines. The reparameterized graphs  $G_1$  and  $G_2$  are a results of two different valid transformations of graph  $G$ . It can be clearly seen that reparameterized graphs  $G_1$  and  $G_2$  have the same st-mincut as graph  $G$ .

will have the same st-mincut. We call these graphs reparameterizations of each other. For any transformation of the energy function which results in such a *reparameterization* we can derive a corresponding transformation for a graph. Under these transformations the resulting graph will be a reparameterization of the original graph and thus will have the same st-mincut. The graph transformations corresponding to energy transformations given by equations (3.12) and (3.13) are shown in Figure 3.5.

The transformations given above are not the only way to obtain a reparameterization. In fact pushing flow through any path in the graph can be seen as performing a valid transformation. The residual graph resulting from this flow is a reparameterization of the original graph where no flow was being passed. This can be easily observed from the fact that the residual graph has the same st-mincut as the original graph, albeit with a different cost. In the next Section we show how the property of graph reparameterization can be used for updating the residual graph when the original graph has been modified and the st-mincut needs to be recomputed.

### 3.4 Recycling Computation

The max-flow solution obtained while minimizing an energy function can be used to efficiently minimize other *similar* energy functions. Consider two energy functions  $E_a$  and  $E_b$  which differ by a few terms. As we have seen in the previous Section, this implies that the graph  $G_b$  representing energy  $E_b$  differs from that representing energy  $E_a$  ( $G_a$ ) by a few edge costs. Suppose we have found the optimal solution of  $E_a$  by solving the max-flow problem on the graph  $G_a$  and now want to find the solution of  $E_b$ . Instead of following the conventional procedure of recomputing the max-flow on  $G_b$  from scratch, we perform the computation by reusing the flows obtained while minimizing  $E_a$ .

Boykov and Jolly [7], in their work on interactive image segmentation used this technique for efficiently recomputing the MAP solution when only the unary terms of the energy function change (due to addition of new hard and soft constraints by the user). However, they did not address the problem of handling changes in the pairwise terms of the energy function which result in changes in the cost of the n-edges of the graph. Our method (explained below) can handle arbitrary changes in the graph.

### 3.4.1 Updating Residual Graphs

The flows through a graph can be used to generate a residual graph (as explained in Section 3.1). Our algorithm works by updating the residual graph obtained from the max-flow computation in graph  $G_a$  to make it represent  $G_b$ . This is done by reducing or increasing the residual capacity of an edge according to the change made to its cost going from  $G_a$  to  $G_b$ .

Recall from equation (3.6) that the flow in an edge of the graph has to satisfy the edge capacity constraint:

$$0 \leq f_{ij} \leq c_{ij} \quad \forall (i, j) \in E. \quad (3.15)$$

While modifying the residual graph, certain flows may violate the new edge capacity constraints (3.15). This is because flow in certain edges might be greater than the capacity of those edges under  $G_b$ . To make these flows consistent with the new edge capacities, we reparameterize the updated graph (using reparameterizations described in the previous Section) to make sure that the flows satisfy the edge capacity constraints (3.15) of the graph. The max-flow is then computed on this reparameterized graph. This gives us the st-mincut solution of graph  $G_b$ , and hence the global minimum solution of energy  $E_b$ .

We now show how the residual graph is transformed to make sure that all edge capacity constraints are satisfied. We use the two graph transformations given in Section 3.3.2 to increase the capacities of edges in  $G_b$  in which the flow exceeds the true capacity. These transformations lead to a reparameterization of the graph  $G_b$ . We can then find the st-mincut on this reparameterized graph to get the st-mincut solution of graph  $G_b$ .

The various changes that might occur to the graph going from  $G_a$  to  $G_b$  can be expressed in terms of changes in the capacity of t-edges and n-edges of the graph. The methods for handling these changes will be discussed now. We use  $c'_{si}$  to refer to the new edge capacity of the edge  $(s, i)$ .  $r'_{si}$  and  $f'_{si}$  are used to represent the updated residual capacity and flow of the edge  $(s, i)$  respectively.

#### Modifying t-edge Capacities

Our method for updating terminal or t-edges is similar to the one used in [7] and is described below. The updated residual capacity of an edge  $(s, i)$  can be computed as:

$$r'_{si} = r_{si} + c'_{si} - c_{si}. \quad (3.16)$$

This can be simplified to:  $r'_{si} = c'_{si} - f_{si}$ . If the flow  $f_{si}$  is greater than the updated edge capacity  $c'_{si}$ , it violates the edge capacity constraint (3.15) resulting in  $r'_{si}$  becoming negative. To make the flow consistent a constant  $\gamma = f_{si} - c'_{si}$  is added to the capacity of both the t-edges  $\{(s,i),(i,t)\}$  connected to the node  $i$ . As has been observed in Section 3.3.2 and in [7], this transformation is an example of graph reparameterization which does not change the minimum cut (its cost changes but not the cut itself). For an illustration see Figure 3.2. The residual capacities thus become:  $r'_{si} = c'_{si} - f_{si} + \gamma = 0$  and,  $r'_{it} = c_{it} - f_{it} + \gamma$ , or  $r'_{ti} = r_{ti} - c'_{si} + f_{si}$ .

## Modifying n-edge Capacities

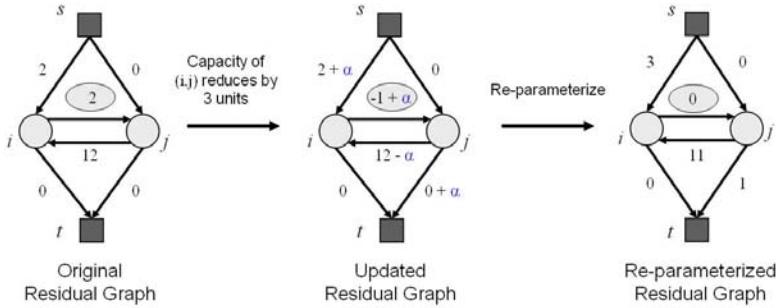
We now describe how the residual graph is updated when n-edge capacities are changed. Observe that updating edge capacities in the residual graph is simple if the new edge capacity  $c'_{ij}$  is greater than or equal to the old edge capacity  $c_{ij}$ . This operation involves addition of extra capacity and thus the flow cannot become inconsistent. The updated residual capacity  $r'_{ij}$  is obtained as:

$$r'_{ij} = r_{ij} + (c'_{ij} - c_{ij}). \quad (3.17)$$

Even if  $c'_{ij}$  is less than  $c_{ij}$ , the procedure still remains trivial if the flow  $f_{ij}$  is less than the new edge capacity  $c'_{ij}$ . This is due to the fact that the reduction in the edge capacity does not affect the flow consistency of the network i.e., flow  $f_{ij}$  satisfies the edge capacity constraint (3.15) for the new edge capacity. The residual capacity of the edge can still be updated according to equation (3.17). The difference in this case is that  $(c'_{ij} - c_{ij})$  is negative and hence will result in the reduction of the residual capacity. In both these cases, the flow through the edge remains unchanged (i.e.,  $f'_{ij} = f_{ij}$ ).

The problem becomes complex when the new edge capacity  $c'_{ij}$  is less than the flow  $f_{ij}$ . In this case,  $f_{ij}$  violates the edge capacity constraint (3.15). To make  $f_{ij}$  consistent, we have to retract the excess flow ( $f_{ij} - c'_{ij}$ ) from the edge  $(i,j)$ . At this point, the reader should note that a trivial solution for this operation would be to push back the flow through the augmenting path it originally came through. However such an operation would be extremely computationally expensive. We now show how we resolve this inconsistency in constant i.e. O(1) time.

The inconsistency arising from excess flow through edge  $(i,j)$  can be resolved by a single valid transformation of the residual graph. This transformation is the same as the one shown in Figure 3.2 for obtaining graph  $G_2$  from  $G$ , and does not change the st-mincut. It leads to a reparameterization of the residual graph which has non-negative residual capacity for the edge  $(i,j)$ . The transformation involves adding a constant  $\alpha = f_{ij} - c'_{ij}$  to the capacity of edges  $(s,i), (i,j)$ , and  $(j,t)$  and subtracting it from the residual capacity of edge  $(j,i)$ . The residual capacity  $r_{ji}$  of edge  $(j,i)$  is greater than the flow  $f_{ij}$  passing through edge  $(i,j)$ . As  $\alpha$  is always



**Fig. 3.6** Restoring consistency using graph reparameterization. The Figure illustrates how edge capacities can be made consistent with the flow by reparameterizing the residual graph. It starts by showing a residual graph consisting of two nodes  $i$  and  $j$  obtained after a max-flow computation. For the second max-flow computation the capacity of edge  $(i, j)$  is reduced by 3 units resulting in the updated residual graph in which the residual capacity of edge  $(i, j)$  is equal to -1. To make the residual capacities positive we reparameterize the graph by adding  $\alpha = 1$  to the capacity of edges  $(i, j)$ ,  $(s, i)$  and  $(j, t)$  and subtracting it from the capacity of edge  $(j, i)$ . This gives us the reparameterized residual graph in which the edge flows are consistent with the edge capacities.

less than  $f_{ij}$  the above transformation does not make the residual capacity of edge  $(j, i)$  negative. The procedure for restoring consistency is illustrated in Figure 3.6.

### 3.4.2 Computational Complexity of Update Operations

In this Section we analyze the computational complexity of various update operations that can be performed on the graph. Modifying an edge cost in the residual graph takes constant time. Arbitrary changes in the graph like addition or deletion of nodes and edges can be expressed in terms of modifying an edge cost. The time complexity of all such changes is  $O(1)$  except for deleting a node where the update time is  $O(k)$ . Here  $k$  is the degree of the node to be deleted.<sup>4</sup>

After the residual graph has been updated to reflect the changes in the energy function, the augmenting path procedure is used to find the maximum flow. This involves repeatedly finding paths with free capacity in the residual graph and saturating them. When no such paths can be found i.e., the source and sink are disconnected in the residual graph, we reach the maximum flow.

The maximum flow from the source to the sink is an upper bound on the number of augmenting paths found by the augmenting path procedure. Also, the total change in edge capacity bounds the increase in the flow  $\nabla f$  defined as:

<sup>4</sup> The capacity of all edges incident on the node has to be made zero which takes  $O(1)$  time per edge.

$$\nabla f \leq \sum_{i=1}^{m'} |c'_{e_i} - c_{e_i}|, \quad \text{where } e_i \in E$$

or,  $\nabla f \leq m' c_{\max}$  where  $c_{\max} = \max(|c'_{e_i} - c_{e_i}|)$ . Thus we get a *loose*  $O(m' c_{\max})$  bound on the number of augmentations, where  $m'$  is the number of edge capacity updates.

### 3.5 Improving Performance by Recycling Search Trees

We have seen how by dynamically updating the residual graph we can reduce the time taken to compute the st-mincut. We can further improve the running time by using a technique motivated by [8].

Typical augmenting path based methods start a new breadth-first search for (source to sink) paths as soon as all paths of a given length are exhausted. For instance, Dinic [16] proposed an augmenting path algorithm which builds search trees to find augmenting paths. This is a computationally expensive operation as it involves visiting almost all nodes of the graph and makes the algorithm slow if it has to be performed too often. To counter this, Boykov and Kolmogorov [8] proposed an algorithm in which they reused the search tree. In their experiments, this new algorithm outperformed the best-known augmenting-path and push-relabel algorithms on graphs commonly used in computer vision.

Motivated from their results we decided to reuse the search trees available from the previous max-flow computation to find the solution in the updated residual graph. This technique saved us the cost of creating a new search tree and made our algorithm substantially faster. The main differences between our algorithm and that of [8] are the presence of the tree restoration stage, and the dynamic selection of active nodes. We will next describe how the algorithm of [8] works and then explain how we modify it to recycle search trees for dynamic graph cuts.

#### 3.5.1 Reusing Search Trees

The algorithm described in [8] maintains two non-overlapping search trees  $S$  and  $T$  with roots at the source  $s$  and the sink  $t$  respectively. In tree  $S$  all edges from each parent node to its children are non-saturated, while in tree  $T$  edges from children to their parents are non-saturated. The nodes that are not in  $S$  or  $T$  are called *free*. The nodes in the search trees  $S$  and  $T$  can be either *active* (can grow by acquiring new children along non-saturated edges) or *passive*. The algorithm starts by setting all nodes adjacent to the terminal nodes as *active*. The three basic stages of the algorithm are as follows:

##### Growth Stage

The search trees  $S$  and  $T$  are grown until they touch each other (resulting in an augmenting path) or all nodes become *passive*. The active nodes explore adjacent

non-saturated edges and acquire new children from the set of free nodes which now become active. As soon as all neighbours of a given active node are explored, the active node becomes passive. When an active node comes in contact with a node from the other tree, an augmenting path is found.

### **Augmentation Stage**

In this stage of the algorithm, flow is pushed through the augmenting path found in the growth stage. This results in some nodes of the trees  $S$  and  $T$  becoming *orphans* since the edges linking them to their parents become saturated. At this point, the source and sink search trees have decomposed into forests.

### **Adoption Stage**

During the adoption stage the search trees are restored by finding a new valid parent (of the same set) through a non-saturated edge for each orphan. If no qualifying parent can be found, the node is made free.

#### **3.5.2 Tree Recycling for Dynamic Graph Cuts**

We now explain our method for recycling search trees of the augmenting path algorithm. Our algorithm differs from that of [8] in the way we initialize the set of active nodes and in the presence of the tree restoration stage.

### **Tree Restoration Stage**

While dynamically updating the residual graph (as explained in Section 3.4) certain edges of the search trees may become saturated and thus need to be deleted. This operation results in the decomposition of the trees into forests and makes certain nodes *orphans*. We keep track of all such edges and before recomputing the st-mincut on the modified residual graph restore the trees by finding a new valid parent for each of them. This process is similar to the adoption stage and is explained below.

The aim of the tree restoration stage is two fold. First to find parents for orphaned nodes, and secondly but more importantly, to make sure that the length of the path from the root node to all other nodes in the tree is as small as possible. This is necessary to reduce the time spent passing flow through an augmenting path. Note that longer augmenting paths would lead to a slower algorithm. This is because the time taken to update the residual capacities of the edges in the augmenting path during the augmentation stage is proportional to the length of the path.

The first objective of the restoration stage can be met by using the adoption stage alone. For the second objective we do the following: Suppose node  $i$  belonged to the source tree before the updates. For each graph node  $i$  which has been affected by the graph updates we check the residual capacities of its t-edges  $((s,i)$  or  $(i,t))$ . We can encounter the following two cases:

1.  $r_{si} \geq r_{it}$  : The original parent of the node (in this case, the source ( $s$ )) is reassigned as the parent of the node.
2.  $r_{si} < r_{it}$  : The parent of the node is changed to the other terminal node ‘sink’ ( $t$ ). This means that the node has now become a member of sink tree  $T$ . All the immediate child nodes of  $i$  are then made orphans as they had earlier belonged to the source tree.

The reassignment of parents of updated nodes according to the above mentioned rules resulted in a moderate but significant improvement in the results.

### **Dynamic Node Activation**

The algorithm of [8] starts by marking the set of all nodes adjacent to the terminal nodes as *active*. This set is usually large and exploring all its constituent nodes is computationally expensive. However this is necessary as an augmenting path can pass through any such node.

In the case of the dynamic st-mincut problem, we can isolate a much smaller subset of nodes which need to be explored for possible augmenting paths. The key observation to be made in this regard is that all new possible augmenting paths are constrained to pass through nodes whose edges have undergone a capacity change. This results in a much smaller active set and makes the max-flow computation significantly faster. When no changes are made to the graph, all nodes in the graph remain *passive* and thus our augmenting path algorithm for computing the max-flow takes no time.

## **3.6 Dynamic Image Segmentation**

The dynamic graph cut algorithm proposed in the previous Section can be used to *dynamically* perform MAP inference in an MRF or CRF. Such an inference procedure is extremely fast and has been used for a number of computer vision problems [34], [10], [25], [54].

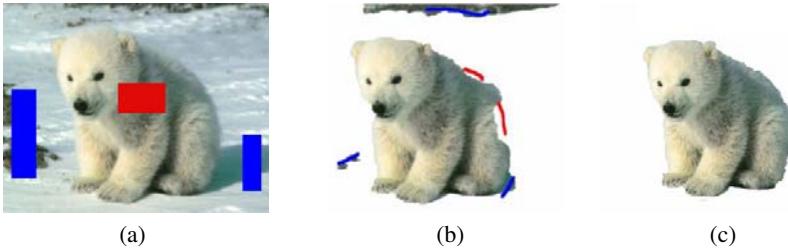
We now describe some applications of dynamic graph cuts. To demonstrate the efficiency of the algorithm, we will provide quantitative results comparing its performance with the dual-search tree algorithm proposed in [8] which has been experimentally shown to be the fastest for several vision problems including image segmentation<sup>5</sup>.

We will call the algorithm of [8] *static* since it starts afresh for each problem instance. The dynamic algorithm which reuses the search trees will be referred to as the *optimized* dynamic graph cut algorithm. It should be noted that while comparing running times the time taken to allocate memory for graph nodes was not considered. Further, to make the experimental results invariant to cache performance we kept the graphs in memory.

Image segmentation has always remained an iconic problem in computer vision. The past few years have seen rapid progress made on it driven by the emergence of

---

<sup>5</sup> For the static algorithm we used the author’s original implementation.



**Fig. 3.7** Interactive image segmentation. The Figure shows how good segmentation results can be obtained using a set of rough region cues supplied by the user. (a) An image with user specified segmentation cues (shown in blue and red). These cues were used to obtain the segmentation shown in image (b). This segmentation is not perfect and can be improved by specifying additional cues which are shown in (b). The final segmentation result is shown in image (c).

powerful optimization algorithms such as graph cuts. Early methods for performing image segmentation worked by coupling colour appearance information about the object and background with the edges present in an image to obtain good segmentations. However, this framework does not always guarantee good results. In particular, it fails in cases where the colour appearance models of the object and background are not discriminative.

A semi-automated solution to this problem was explored by Boykov and Jolly [7] in their work on interactive image segmentation. They showed how users could refine segmentation results by specifying additional constraints. This can be done by labelling particular regions of the image as ‘object’ or ‘background’ and then computing the MAP solution of the CRF again. The interactive image segmentation process is illustrated in Figure 3.7.

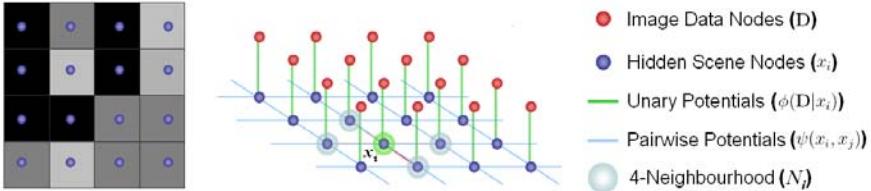
### 3.6.1 CRFs for Image Segmentation

The image segmentation problem is commonly formulated using the CRF model described in Section 3.1. In the context of image segmentation, the vertex set  $\mathcal{V}$  corresponds to the set of all image pixels,  $\mathcal{N}$  is a neighbourhood defined on this set<sup>6</sup>, the set  $\mathcal{L}$  consists of the labels representing the different image segments (which in our case are ‘foreground’ and ‘background’), and the value  $x_v$  denotes the labelling of the pixel  $v$  of the image. Every configuration  $\mathbf{x}$  of such a CRF defines a segmentation. The image segmentation problem can thus be solved by finding the least energy configuration of the CRF.

The energy function characterizing the CRFs used for image segmentation can be written as a sum of likelihood ( $\phi(\mathbf{D}|x_i)$ ) and prior ( $\psi(x_i, x_j)$ ) terms as:

---

<sup>6</sup> In this work, we have used the standard 8-neighbourhood i.e., each pixel is connected to the 8 pixels surrounding it.



**Fig. 3.8** The pairwise MRF commonly used to model image labelling problems. The random field contains a hidden node corresponding to each pixel in the image. The MRF shown in the figure has a 4-neighbourhood, i.e., each node representing the random variables is connected to 4 neighbouring nodes.

$$\Psi_l(\mathbf{x}) = \sum_{i \in \mathcal{V}} \left( \phi(\mathbf{D}|x_i) + \sum_{j \in \mathcal{N}_i} \psi(x_i, x_j) \right) + \text{const.} \quad (3.18)$$

The term  $\phi(\mathbf{D}|x_i)$  in the CRF energy is the data log likelihood which imposes individual penalties for assigning any label  $k \in \mathcal{L}$  to pixel  $i$ . If we only take the appearance model into consideration, the likelihood is given by

$$\phi(\mathbf{D}|x_i) = -\log p(i \in \mathcal{S}_k | \mathcal{H}_k) \quad \text{if } x_i = k, \quad (3.19)$$

where  $\mathcal{H}_k$  is the RGB (or for grey scale images, the intensity value) distribution for the segment  $\mathcal{S}_k$  denoted by label  $k \in \mathcal{L}$ <sup>7</sup>. The probability of a pixel belonging to a particular segment i.e.  $p(i \in \mathcal{S}_k | \mathcal{H}_k)$  is proportional to the likelihood  $p(I_i | \mathcal{H}_k)$ , where  $I_i$  is the colour intensity of the pixel  $i$ . The likelihood  $p(I_i | \mathcal{H}_k)$  is generally computed from the colour histogram of the pixels belonging to the segment  $\mathcal{S}_k$ .

The prior  $\psi(x_i, x_j)$  terms takes the form of a Generalized Potts model:

$$\psi(x_i, x_j) = \begin{cases} K_{ij} & \text{if } x_i \neq x_j, \\ 0 & \text{if } x_i = x_j. \end{cases} \quad (3.20)$$

The CRF used to model the image segmentation problem also contains a contrast term which favours pixels with similar colours having the same label [5][7]. This term is incorporated in the energy function by increasing the cost within the Potts model (for two neighbouring variables being different) in proportion to the similarity in intensities of their corresponding pixels. In our experiments, we use the function:

$$\gamma(i, j) = \lambda \exp \left( \frac{-g^2(i, j)}{2\sigma^2} \right) \frac{1}{\text{dist}(i, j)}, \quad (3.21)$$

where  $g^2(i, j)$  measures the difference in the RGB values of pixels  $i$  and  $j$  and  $\text{dist}(i, j)$  gives the spatial distance between  $i$  and  $j$ . This is a likelihood term (not

---

<sup>7</sup> In our problem, we have only 2 segments i.e., the foreground and the background.

prior) as it is based on the data, and hence has to be added separately from the smoothness prior. The energy function of the CRF now becomes

$$\Psi_2(\mathbf{x}) = \sum_{i \in \mathcal{V}} \left( \phi(\mathbf{D}|x_i) + \sum_{j \in \mathcal{N}_i} (\phi(\mathbf{D}|x_i, x_j) + \psi(x_i, x_j)) \right) \quad (3.22)$$

The contrast term of the energy function has the form

$$\phi(\mathbf{D}|x_i, x_j) = \begin{cases} \gamma(i, j) & \text{if } x_i \neq x_j \\ 0 & \text{if } x_i = x_j. \end{cases} \quad (3.23)$$

By adding this term to the energy, we have diverged from the strict definition of an MRF. The resulting energy function in fact now characterizes a Conditional Random Field [42]. The pairwise MRF commonly used to model image labelling problems is shown in Figure 3.8.

### 3.6.2 Image Segmentation in Videos

The object-background segmentation problem aims to cut out user specified objects in an image [7]. We consider the case when this process has to be performed over all frames in a video sequence. The problem is formulated as follows.

The user specifies hard and soft constraints on the segmentation by providing segmentation cues or seeds on only the first frame of the video sequence. The *soft constraints* are used to build colour histograms for the *object* and *background*. These histograms are later used for calculating the likelihood term  $\phi(\mathbf{D}|x_i)$  of the energy function (3.22) of the CRF. This is done for all the frames of the video sequence.

The *hard constraints* are used for specifying pixel positions which are constrained to take a specific label (*object* or *background*) in all the frames of the video sequence. Note that unlike soft constraints, the pixel positions specified under hard constraints do not contribute in the construction of the colour histograms for the *object* and *background*. This is different from the user-input strategy adopted in [7]. In our method the hard constraints are imposed on the segmentation by incorporating them in the likelihood term  $\phi(\mathbf{D}|x_i)$ . This is done by imposing a very high cost for a label assignment that violates the hard constraints in a manner similar to [7]. This method for specifying hard constraints has been chosen because of its simplicity. Readers should refer to [73] for a sophisticated method for specifying hard constraints for the video segmentation problem. Figure 3.9 demonstrates the use of constraints in the image segmentation process. The segmentation results are shown in Figure 3.10.

### 3.6.3 Experimental Results

In this Section we demonstrate the performance of our dynamic graph cut algorithm on the image segmentation problem. We compare the time taken by our algorithm with that needed by the algorithm proposed in [8].



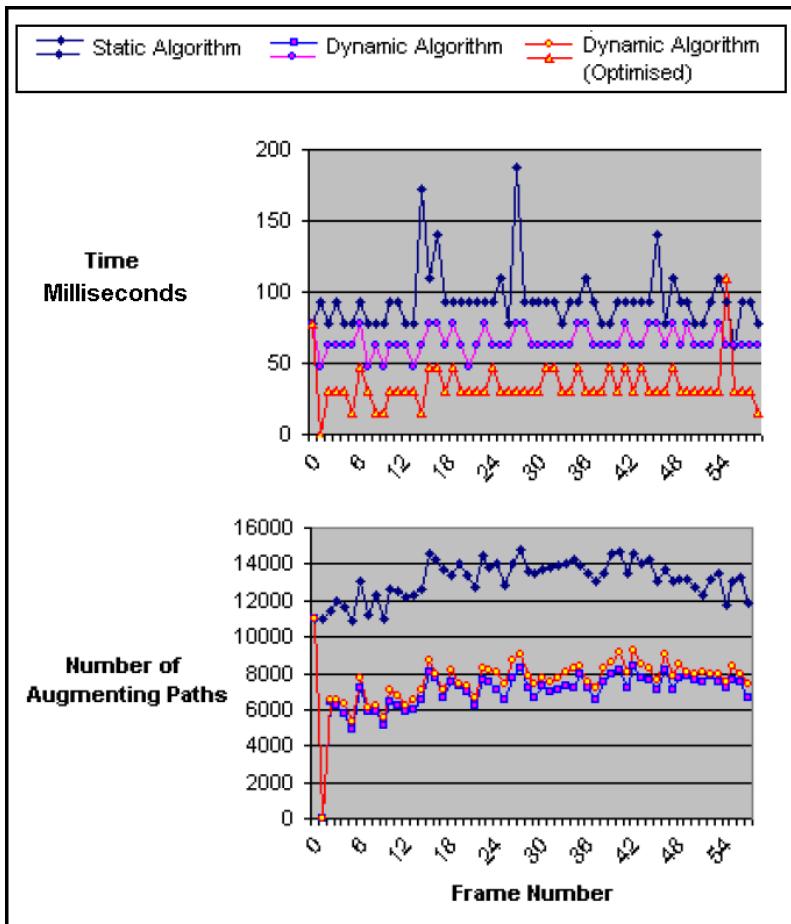
**Fig. 3.9** Segmentation in videos using user seeds. The first image shows one frame of the input video with user segmentation seeds (the black and white boxes). The image pixels contained in these boxes are used to learn histograms modelling foreground and background likelihoods. The second image shows the segmentation result obtained using these likelihoods with the method of [7]. The result contains a certain portion of the background wrongly marked as the foreground due to similarity in colour. This error in the segmentation can be removed by the user by specifying a hard constraint. This involves marking a set of pixel positions in the wrongly labelled region as background (shown as the checkered region in the second image). This constraint is used for all the frames of the video sequence. The third image is the final segmentation result.



**Fig. 3.10** Segmentation results of the human lame walk video sequence.

In the interactive image segmentation experiments, we observed that dynamic graph cuts resulted in a massive improvement in the running time. For the image shown in Figure 3.7, the time taken by the static st-mincut algorithm to compute the refined solution (from scratch) was 120 milliseconds. The dynamic algorithm computed the same solution in 45 milliseconds, while the dynamic (optimized) algorithm only required 25 milliseconds.

We now discuss the results of image segmentation in videos. The video sequences used in our tests had between one hundred to a thousand image frames. For all the video sequences dynamically updating the residual graph produced a decrease in the number of augmenting paths. Further, the dynamic algorithms (normal and optimized) were substantially faster than the *static* algorithm. The average



**Fig. 3.11** Running time and number of augmenting paths found by static and dynamic st-mincut algorithms. Observe that as the first and second frames of the video sequence are the same, the residual graph does not need to be updated, which results in no augmenting paths found by the dynamic algorithms when segmenting frame 2. Further, the optimized dynamic algorithm takes no time for computing the segmentation for the second image frame as the CRFs corresponding to the first and second image frames are the same and thus no modifications were needed in the residual graph and search trees. However, the normal dynamic algorithm takes a small amount of time since it recreates the search trees for every problem instance from scratch.

running times per image frame for the static, dynamic and optimized-dynamic algorithms for the human lame walk sequence<sup>8</sup> of size 368x256 were 91.4, 66.0, and 33.6 milliseconds and for the grazing cow sequence of size 720x578 were 188.8, 151.3, and 78.0 milliseconds respectively. The time taken by the dynamic algorithm

<sup>8</sup> Courtesy Derek Magee, University of Leeds.

includes the time taken to recycle the search trees. The experiments were performed on a Pentium 4 2.8 GHz machine.

The graphs in Figure 3.11 show the performance of the algorithms on the first sixty frames of the human lame walk sequence. Observe that the number of augmenting paths found is lowest for the dynamic algorithm, followed by the dynamic (optimized) and then the static algorithm. The use of more augmenting paths by the dynamic (optimized) algorithm is due to the utilization of recycled search trees which produce long augmenting paths.

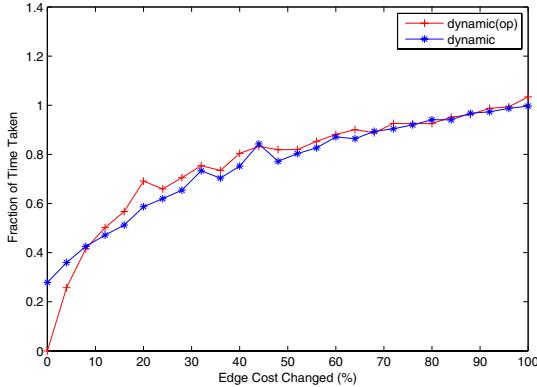
### 3.6.4 Reusing Flow vs. Reusing Search Trees

In this Section, the relative contributions of reusing flow and search trees in improving the running time of the dynamic algorithm are discussed.

The procedure for constructing a search tree has linear time complexity and thus should be quite fast. Further as seen in Figure 3.11 using a fresh search tree after every graph update results in fewer augmenting paths. From these results it might appear that recycling search trees would not yield a significant improvement in running time. However this is not the case in practice as seen in Figure 3.12. This is because although the complexity of search tree construction is linear in the number of edges in the graph, the time taken for tree construction is still substantial. This is primarily due to the nature of graphs used in computer vision problems. The number of nodes/edges in these graphs may be of the order of millions. For instance, when segmenting an image of size  $640 \times 480$ , max-flow on a graph consisting of roughly  $3 \times 10^5$  nodes and more than 2 million edges needs to be computed. The total time taken for this operation is 90 milliseconds (msec) out of which almost 15 msec is spent on constructing the search tree.

The time taken by the dynamic algorithm to compute the st-mincut decreases with the decrease in the number of changes made to the graph. However, as the time taken to construct the search tree is independent of the number of changes, it remains constant. This results in a situation where if only a few changes to the graph are made (as in the case of min-marginal computation [34]), the dominant part of computation time is spent on constructing the search tree itself. By reusing search trees we can get rid of this constant cost of creating a search tree and replace it with a change dependent tree restoration cost.

The exact amount of speed-up contributed by reusing flow and search trees techniques varies with the problem instance. For a typical interactive image segmentation example, the first st-mincut computation takes 120 msec out of which 30 msec is spent on constructing the search tree. We need to recompute the st-mincut after further user interaction (which results in changes in the graphs). For the later st-mincut computation, if we construct a new search tree then the time taken by the algorithm is 45 msec (a speed up of roughly 3 times) out of which 30 msec is used for tree creation and 15 msec is used for flow computation. However, if we use reuse the search trees, then the algorithm takes only 25 msec (a speed up of 5



**Fig. 3.12** Behavior of the dynamic algorithm. The Figure illustrates how the time taken by the dynamic algorithm (with/without tree recycling) changes with the number of modifications made to the graph. The graph shows the fraction of time taken to compute the st-mincut in the updated residual graph (with/without tree recycling) compared to that taken for computing the st-mincut in the original graph using the algorithm of [8]. For this experiment, we used a graph consisting of  $1 \times 10^5$  nodes which were connected in a 8-neighbourhood. The dynamic algorithm with tree recycling is referred as dynamic(op).

times) out of which 7 msec is used for recycling the tree and 18 msec is used for flow computation.

Our results indicate that when a small number of changes are made to the graph the recycled search tree works quite well in obtaining short augmenting paths. The time taken for recycling search trees is also small compared to the time taken to create a new search tree in a large graph. With increased change in the graph the advantage in using the recycled search tree fades due to the additional number of flow augmentations needed as a result of longer augmentation paths obtained from the search tree.

### 3.7 Simultaneous Segmentation and Pose Estimation of Humans

In this Section we present a novel algorithm for performing integrated segmentation and 3D pose estimation of a human body from multiple views. Unlike other state of the art methods which focus on either segmentation or pose estimation individually, our approach tackles these two tasks together. Our method works by optimizing a cost function based on a Conditional Random Field (CRF). This has the advantage that all information in the image (edges, background and foreground appearances), as well as the prior information on the shape and pose of the subject can be combined and used in a Bayesian framework. Optimizing such a cost function would have been computationally infeasible earlier. However, our recent research in dynamic graph

cuts allows this to be done much more efficiently than before. We demonstrate the efficacy of our approach on challenging motion sequences. Although we target the human pose inference problem in this work, our method is completely generic and can be used to segment and infer the pose of any rigid, deformable or articulated object.

Human pose inference is an important problem in computer vision. It stands at the crossroads of various important applications ranging from Human Computer Interaction (HCI) to surveillance. The importance and complexity of this problem can be gauged by observing the number of papers which have tried to deal with it [1, 18, 31, 57, 23, 58, 60, 68, 43, 15, 46, 51]. Most algorithms which perform pose estimation require the segmentation of humans as an essential introductory step [1, 31, 57]. This precondition limits the use of these techniques to scenarios where good segmentations are made available by enforcing strict studio conditions like blue-screening. Otherwise a preprocessing step must be performed in an attempt to segment the human, such as [62]. These approaches however cannot obtain good segmentations in challenging scenarios which have: complex foreground and background, multiple objects in the scene, and moving camera/background. Some pose inference methods exist which do not need segmentations. These rely on features such as chamfer distance [23], appearance [58], or edge and intensity [60]. However, none of these methods is able to efficiently utilize all the information present in an image, and fail if the feature detector they are using fails. This is partly because the feature detector is not coupled to the knowledge of the pose and nature of the object to be segmented.

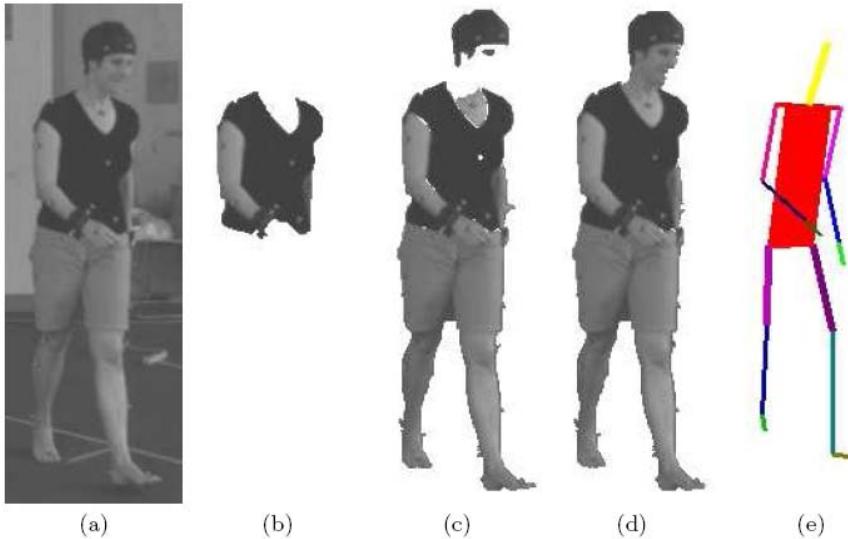
The question is then, how to simultaneously obtain the segmentation and human pose using all available information contained in the images?

Some elements of the answer to this question have been described by Kumar *et al.* [40]. Addressing the object segmentation problem, they report that “*samples from the Gibbs distribution defined by a Markov Random Field very rarely give rise to realistic shapes*”. As an illustration of this statement, Figure 3.13(b) shows the segmentation result corresponding to the maximum a posteriori (MAP) solution of the Conditional Random Field (CRF) incorporating information about the image edges and appearances of the object and background. It can be clearly seen that this result is nowhere close to the ground truth.

### Shape Priors and Segmentation

In recent years, a number of papers have tried to couple MRFs or CRFs used for modelling the image segmentation problem, with information about the nature and shape of the object to be segmented [40, 26, 21, 78]. One of the earliest methods for combining MRFs with a shape prior was proposed by Huang *et al.* [26]. They incrementally found the MAP solution of an extended MRF<sup>9</sup> integrated with a probabilistic deformable model. They were able to obtain a refined estimate of the object

<sup>9</sup> It is named an *extended* MRF due to the presence of an extra layer in the MRF to cope with the shape prior.



**Fig. 3.13** Improving segmentation results by incorporating more information in the CRF. (a) Original image. (b) The segmentation obtained corresponding to the MAP solution of a CRF consisting of colour likelihood and contrast terms as described in [2]. We give the exact formulation of this CRF in Section 3.6.1 (c) The result obtained when the likelihood term of the CRF also takes into account the Gaussian Mixture Models (GMM) of individual pixel intensities as described in Section 3.7.1 (d) Segmentation obtained after incorporating a ‘pose-specific’ shape prior in the CRF as explained in Section 3.7.1. The prior is represented as the distance transform of a stickman which guarantees a human-like segmentation. (e) The stickman model after optimization of its 3D pose (see Section 3.7.2). Observe how incorporating the individual pixel colour models in the CRF (c) gives a considerably better result than the one obtained using the standard appearance and contrast based representation (b). However the segmentation still misses the face of the subject. The incorporation of a stickman shape prior ensures a human-like segmentation (d) and provides simultaneously (after optimization) the 3D pose of the subject (e).

contour by using belief propagation in the area surrounding the contour of this deformable model. This process was iterated till convergence.

The problem however was still far from being completely solved since objects in the real world change their shapes constantly and hence it is difficult to ascertain what would be a good choice for a prior on the shape. This complex and important problem was addressed by the work of Kumar *et al.* [40]. They modelled the segmentation problem by combining CRFs with layered pictorial structures (LPS) which provided them with a realistic shape prior described by a set of latent shape parameters. Their cost function was a weighted sum of the energy terms for different shape parameters (samples). The weights of this energy function were obtained by using the Expectation-Maximization (EM) algorithm. During this optimization procedure, a graph cut had to be computed in order to obtain the segmentation score each time

any parameter of the CRF was changed. This made their algorithm extremely computationally expensive.

Although their approach produced good results, it had some shortcomings. It was focused on obtaining good segmentations and did not provide the pose of the object explicitly. Moreover, a lot of effort had to be spent to learn the exemplars for different parts of the LPS model. Recently, Zhao and Davis [78] exploited the idea of object-specific segmentation to improve object recognition and detection. Their method worked by coupling the twin problems of object detection and segmentation in a single framework. They matched exemplars to objects in the image using chamfer matching and thus like [40] also suffered from the problem of maintaining a huge exemplar set for complex objects. We will describe how we overcome the problem of maintaining a huge exemplar set by using a simple articulated stickman model, which is not only efficiently renderable, but also provides a robust human-like segmentation and accurate pose estimate. To make our algorithm computationally efficient we use the dynamic graph cut algorithm.

### **Shape Priors in Level Sets**

Prior knowledge about the shape to be segmented has also been used in level set methods for obtaining an object segmentation. Like [40] these methods learn the prior using a number of training shapes. Leventon *et al.* [45] performed principal component analysis on these shapes to get an embedding function which was integrated in the evolution equation of the level set. More recently, Cremers *et al.* [13] have used kernel density estimation and intrinsic alignment to embed more complex shape distributions. Compared to [40] and [78] these methods have a more compact representation of the shape prior. However, they suffer from the disadvantage that equations for level set evolution may lead to a local minima.

### **Human Pose Estimation**

In the last few years, several techniques have been proposed for tackling the pose inference problem. In particular, the works of Agarwal and Triggs [1] using relevance vector machines and that of Shakhnarovich *et al.* [57] based on parameter sensitive hashing induced a lot of interest and have been shown to give good results. Some methods for human pose estimation in monocular images use a tree-structured model to capture the kinematic relations between parts such as the torso and limbs [46, 18, 51]. They then use efficient inference algorithms to perform exact inference in such models. In their recent work, Lan and Huttenlocher [43] show how the tree-structured restriction can be overcome while not greatly increasing the computational cost of estimation.

### **Overview of the Method**

Our method does not require a feature extraction step but uses all the data in the image. We formulate the problem in a Bayesian framework building on the object-specific CRF [40] and provide an efficient method for its solution called POSECUT.

We include a human *pose-specific* shape prior in the CRF used for image segmentation, to obtain high quality segmentation results. We refer to this integrated model as a *pose-specific* CRF. Unlike Kumar *et al.* [40], our approach does not require the laborious process of learning exemplars. Instead we use a simple articulated stickman model, which together with an CRF is used as our shape prior. The experimental results show that this model suffices to ensure human-like segmentations.

Given an image, the solution of the pose-specific CRF is used to measure the quality of a 3D body pose. This cost function is then optimized over all pose parameters using dynamic graph cuts to provide both an object-like segmentation and the pose. The astute reader will notice that although we focus on the human pose inference problem, our method is in-fact general and can be used to segment and/or infer the pose of any object. We believe that our methodology is completely novel and we are not aware of any published methods which perform simultaneous segmentation and pose estimation. To summarize, the novelties of our approach include:

- An efficient method for combined object segmentation and pose estimation (POSECUT).
- Integration of a simple ‘stickman prior’ based on the skeleton of the object in a CRF to obtain a *pose-specific* CRF which helps us in obtaining high quality object pose estimate and segmentation results.

### 3.7.1 Pose Specific CRF for Image Segmentation

The CRF framework for image segmentation described in Section 3.6.1 uses likelihood terms which are only based on the pixel colour. This term is quite weak and thus does not always guarantee good results. In particular, it fails in cases where the colour appearance models of the object and background are not discriminative as seen in Figure 3.13(b). The problem becomes even more pronounced in the case of humans where we have to deal with the various idiosyncrasies of human clothing.

From the work of Boykov and Jolly [7] on interactive image segmentation we made the following interesting observations:

- *Simple user supplied shape cues used as rough priors for the object segmentation problem produced excellent results.*
- *The exact shape of the object can be induced from the edge information embedded in the image.*

Taking these into consideration, we hypothesized that the accurate exemplars used in [40] to generate shape priors were in-fact an overkill and could be replaced by much simpler models. Motivated by these observations we decided against using a sophisticated shape prior. We have used two simple models in our work which are described below.

#### Stickman Model

We used a simple articulated stickman model for the full body human pose estimation problem. The model is shown in Figure 3.13(e). It is used to generate a rough

pose-specific shape prior on the segmentation. As can be seen from the segmentation results in Figure 3.13(d), the stickman model helped us to obtain excellent segmentation results. The model has 26 degrees of freedom consisting of parameters defining absolute position and orientation of the torso, and the various joint angle values. There were no constraints or joint-limits incorporated in our model.

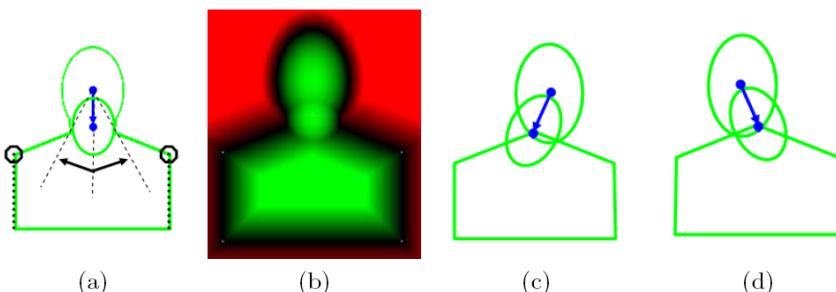
## The Upper body Model

The second model was primarily designed for the problem of segmenting the human speaker in video conference scenarios. The model can be seen in Figure 3.14. It is parameterized by 6 parameters which encode the  $x$  and  $y$  location of the two shoulders and the length and angle of the neck.

We now describe how the image segmentation problem can be modeled using a *pose-specific CRF*. Our pose specific CRF is obtained by augmenting the conventionally used CRF model for image segmentation (see Section 3.6.1) with potentials based on the shape of the object to be segmented, and appearances of individual pixels.

### *Modeling Pixel Intensities by Gaussian Mixture Models*

The CRF defined in Section 3.6.1 performs poorly when segmenting images in which the appearance models of the foreground and background are not highly discriminative. When working on video sequences, we can use a background model developed using the Grimson-Stauffer [62] algorithm to improve our results. This algorithm works by representing the colour distribution of each pixel position in the video as a Gaussian Mixture Model (GMM). The likelihoods of a pixel for being background or foreground obtained by this technique are integrated in our CRF. Figure 3.13(c) shows the segmentation result obtained after incorporating this information in our CRF formulation.



**Fig. 3.14** The human upper body model. (a) The human upper body model parameterized by 6 parameters encoding the  $x$  and  $y$  location of the two shoulders, the length of the neck, and the angle of the neck with respect to the vertical. (b) The shape prior generated using the model. Pixels more likely to belong to the foreground/background are green/red. (c) and (d) The model rendered in two poses.

### Incorporating the Pose-specific Shape Prior

Though the results obtained from the above formulation look decent, they are not perfect. Note that there is no prior on the segmentation to look human like. Intuitively, incorporating such a constraint in the CRF would improve the segmentation. In our case, this prior should be *pose-specific* as it depends on what pose the object (the human) is in. Kumar *et. al.* [40], in their work on interleaved object recognition and segmentation, used the result of the recognition to develop a shape prior over the segmentation. This prior was defined by a set of latent variables which favoured segmentations of a specific pose of the object. They called this model the Object Category Specific CRF, which had the following energy function:

$$\Psi_3(\mathbf{x}, \omega) = \sum_i (\phi(\mathbf{D}|x_i) + \phi(x_i|\omega)) + \sum_j (\phi(\mathbf{D}|x_i, x_j) + \psi(x_i, x_j)) \quad (3.24)$$

with posterior  $p(\mathbf{x}, \omega|\mathbf{D}) = \frac{1}{Z_3} \exp(-\Psi_3(\mathbf{x}, \omega))$ . Here  $\omega \in \mathbb{R}_p$  is used to denote the vector of the object pose parameters. The shape-prior term of the energy function for a particular pose of the human is shown in Figure 3.15(e). This is a distance transform generated from the stick-man model silhouette using the fast implementation of Felzenszwalb and Huttenlocher [17].

The function  $\phi(x_i|\omega)$  was chosen such that given an estimate of the location and shape of the object, pixels falling near to that shape were more likely to be labelled as ‘foreground’ and vice versa. It has the form:  $\phi(x_i|\omega) = -\log p(x_i|\omega)$ . We follow the formulation of [40] and define  $p(x_i|\omega)$  as

$$p(x_i = \text{figure}|\omega) = 1 - p(x_i = \text{ground}|\omega) = \frac{1}{1 + \exp(\mu * (d(i, \omega) - d_r))}, \quad (3.25)$$

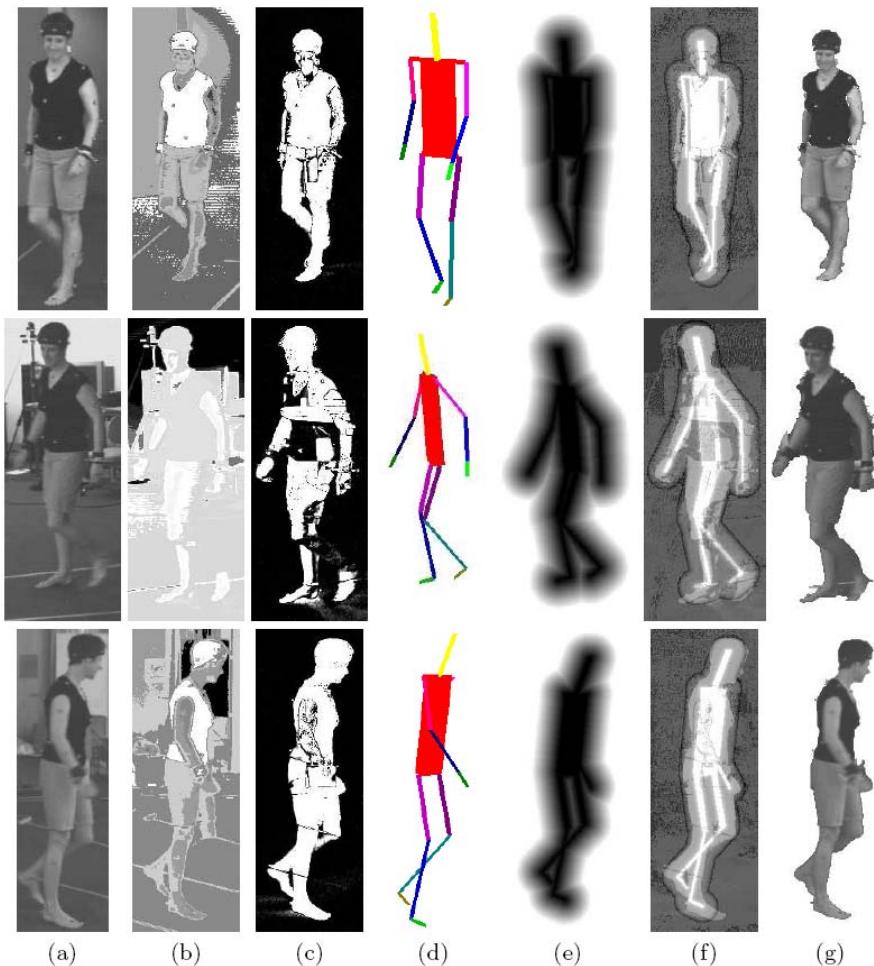
where  $d(i, \omega)$  is the distance of a pixel  $i$  from the shape defined by  $\omega$  (being negative if inside the shape). The parameter  $d_r$  decides how ‘fat’ the shape should be, while parameter  $\mu$  determines the ratio of the magnitude of the penalty that points outside the shape have to face, compared to the points inside the shape.

### Inference in the CRF Using Graph Cuts

Recall from Section 3.1 that energy functions like the one defined in (3.24) can be solved using graph cuts if they are *sub-modular* [32]. A function  $f : \{0, 1\}^n \rightarrow \mathbb{R}$  is submodular if and only if all its projections on 2 variables ( $f^p : \{0, 1\}^2 \rightarrow \mathbb{R}$ ) satisfy:

$$f^p(0, 0) + f^p(1, 1) \leq f^p(0, 1) + f^p(1, 0). \quad (3.26)$$

For the pairwise potentials, this condition can be seen as implying that the energy for two labels taking similar values should be less than the energy for them taking different values. In our case, this is indeed the case and thus we can find the optimal configuration  $\mathbf{x}^* = \min_{\mathbf{x}} \Psi_3(\mathbf{x}, \omega)$  using a single graph cut. The labels of the latent variable in this configuration give the segmentation solution.



**Fig. 3.15** Different terms of our pose specific CRF. (a) Original image. (b) The ratios of the likelihoods of pixels being labelled foreground/background ( $\phi(\mathbf{D}|\mathbf{x}_i = \text{'fg'}) - \phi(\mathbf{D}|\mathbf{x}_i = \text{'bg'})$ ). These values are derived from the colour intensity histograms. (c) The segmentation results obtained by using the GMM models of pixel intensities. (d) The stickman in the optimal pose (see Sections 3.7.1 and 3.7.2). (e) The shape prior (distance transform) corresponding to the optimal pose of the stickman. (f) The ratio of the likelihoods of being labelled foreground/background using all the energy terms (colour histograms defining appearance models, GMMs for individual pixel intensities, and the pose-specific shape prior (see Sections 3.6.1 [3.7.1] and 3.7.1))  $\Psi_3(x_i = \text{'fg'}, \omega) - \Psi_3(x_i = \text{'bg'}, \omega)$ . (g) The segmentation result obtained from our algorithm which is the MAP solution of the energy  $\Psi_3$  of the pose-specific CRF.

### 3.7.2 Formulating the Pose Inference Problem

Since the segmentation of an object depends on its estimated pose, we would like to make sure that our shape prior reflects the actual pose of the object. This takes us to our original problem of finding the pose of the human in an image. In order to solve this, we start with an initial guess of the object pose and optimize it to find the correct pose. When dealing with videos, a good starting point for this process would be the pose of the object in the previous frame. However, more sophisticated methods could be used based on object detection [63] at the expense of increasing the computation time.

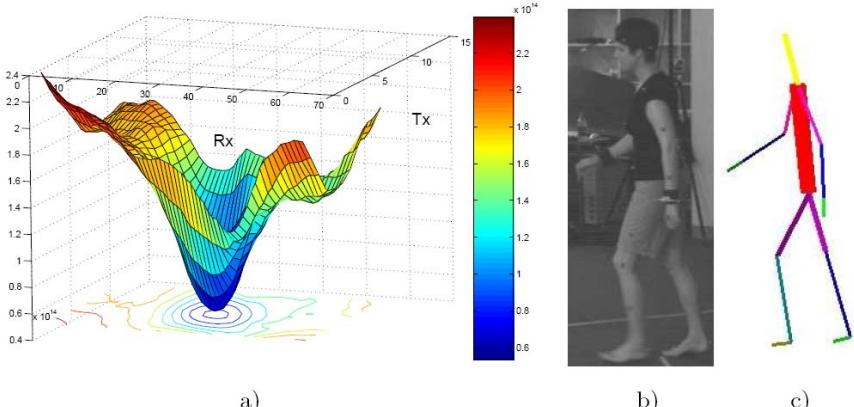
One of the key contributions of this work is to show how given an image of the object, the pose inference problem can be formulated in terms of an optimization problem over the CRF energy given in (3.24). Specifically, we solve the problem:

$$\omega_{\text{opt}} = \arg \min_{\omega, \mathbf{x}} \Psi_3(\mathbf{x}, \omega). \quad (3.27)$$

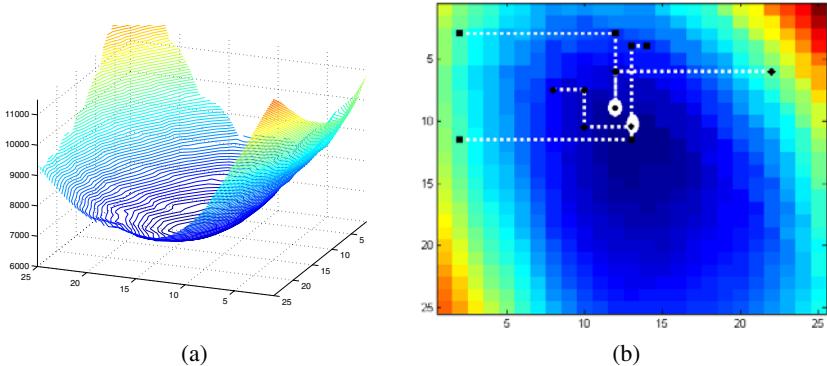
The minimization problem defined above contains both discrete ( $\mathbf{x} \in \{0, 1\}^n$ ) and continuous ( $\omega \in \mathbb{R}_P$ ) valued variables and thus is a mixed integer programming problem. The large number of variables involved in the energy function  $\Psi_3(\mathbf{x}, \omega)$  make it especially challenging to minimize. To solve the minimization problem (3.27), we decompose it as:  $\omega_{\text{opt}} = \arg \min_{\omega} \mathcal{F}(\omega)$ , where

$$\mathcal{F}(\omega) = \min_{\mathbf{x}} \Psi_3(\mathbf{x}, \omega). \quad (3.28)$$

For any value of  $\omega$ , the function  $\Psi_3(\mathbf{x}, \omega)$  is submodular in  $\mathbf{x}$  and thus can be minimized in polynomial time by solving a single st-mincut problem to give the value of  $\mathcal{F}(\omega)$ .



**Fig. 3.16** Inferring the optimal pose. a) The values of  $\min_{\mathbf{x}} \Psi_3(\mathbf{x}, \omega)$  obtained by varying the global translation and rotation of the shape prior in the x-axis. b) Original image. c) The pose obtained corresponding to the global minimum of the energy.



**Fig. 3.17** Optimizing the pose parameters. (a) The values of  $\min_{\mathbf{x}} \Psi_3(\mathbf{x}, \omega)$  obtained by varying the rotation and length parameters of the neck. (b) The image shows five runs of the Powell minimization algorithm [49] which are started from different initial solutions.

We will now explain how we minimize  $\mathcal{F}(\omega)$  to get the optimal value of the pose parameters. Figure 3.16 shows how the function  $\mathcal{F}(\omega)$  depends on parameters encoding the rotation and translation of our stickman model in the x-axes. It can be seen that the function surface is unimodal in a large neighbourhood of the optimal solution. Hence, given a good initialization of the pose  $\omega$ , it can be reliably optimized using any standard optimization algorithm like gradient descent. In our experiments, we used the Powell minimization [49] algorithm for optimization.

Figure 3.17(a) shows how the function  $\mathcal{F}(\omega)$  changes with changes to the neck angle and length parameters of the upper body model shown in Figure 3.14. Like in the case of the 3D stickman model, the energy surface is well behaved near the optimal pose parameters. Our experiments showed that the Powell minimization algorithm is able to converge to almost the same point for different initializations (see Figure 3.17(b)).

### Failure Modes

It can be seen that the function  $\mathcal{F}(\omega)$  is not unimodal over the whole domain and contains local minima. This multi-modality of  $\mathcal{F}(\omega)$  can cause a gradient descent algorithm to get trapped and converge to a local minimum. In our experiments we observed that these spurious minima lie quite far from the globally optimal solution. We also observed that the pose of the human subject generally does not change substantially from one frame to the next. This lead us to use the pose estimate from the previous frame as an initialization for the current frame. This good initialization for the pose estimate made sure that spurious minima do not effect our method.

The failure rate of our method can be further improved by using object detection systems which provide a better initialization of the pose of the object. Scenarios where the method still converges to a local minima can be detected and dealt with

using the strategy discussed in Section 3.7.5 which was used in our recent work on object detection and segmentation [52].

### Resolving Ambiguity Using Multiple Views

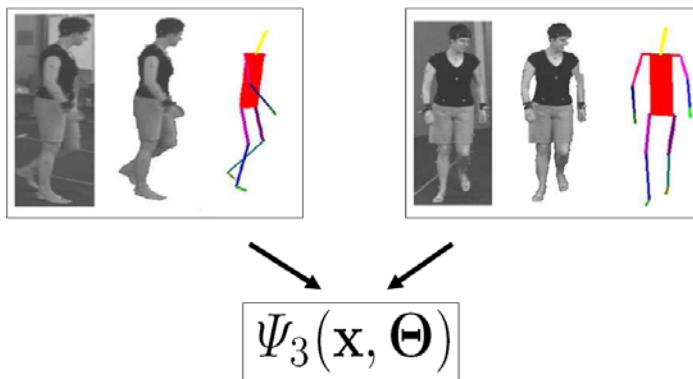
The human pose inference problem in the context of monocular images suffers from ambiguity. This is because of the one-many nature of the mapping that relates a human shape as seen in an image and the corresponding human pose. In other words, many possible poses can explain the same human shape. This ambiguity can be resolved by using multiple views of the object ('human'). Our framework has the advantage that information from multiple views can be integrated into a single optimization framework. Specifically, when dealing with multiple views we solve the problem:

$$\omega_{\text{opt}} = \arg \min_{\omega} \left( \sum_{\text{Views}} \min_{\mathbf{x}} (\Psi_3(\mathbf{x}, \omega)) \right). \quad (3.29)$$

The framework is illustrated in Figure 3.18.

### Dynamic Energy Minimization Using Graph Cuts

As explained earlier global minima of energies like the one defined in (3.24) can be found by graph cuts [28]. The time taken for computing a graph cut for a reasonably sized CRF is of the order of seconds. This would make our optimization algorithm extremely slow since we need to compute the global optimum of  $\Psi_3(\mathbf{x}, \omega)$  with respect to  $\mathbf{x}$  multiple number times for different values of  $\omega$ . The graph cut computation can be made significantly faster by using the dynamic graph cut algorithm proposed in Section 3.3. This algorithm works by using the solution of the previous



**Fig. 3.18** Resolving ambiguity in pose using multiple views. The Figure shows how information from different views of the human can be integrated in a single energy function, which can be used to find the true pose of the human subject.

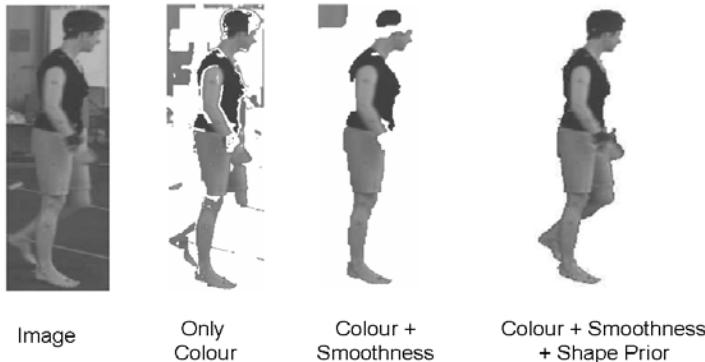
graph cut computation for solving the new instance of the problem. We obtained a speed-up in the range of 15-20 times by using the dynamic graph cut algorithm.

### 3.7.3 Experiments

We now discuss the results obtained by our method. We provide the segmentation and pose estimation results individually.

#### Segmentation Results

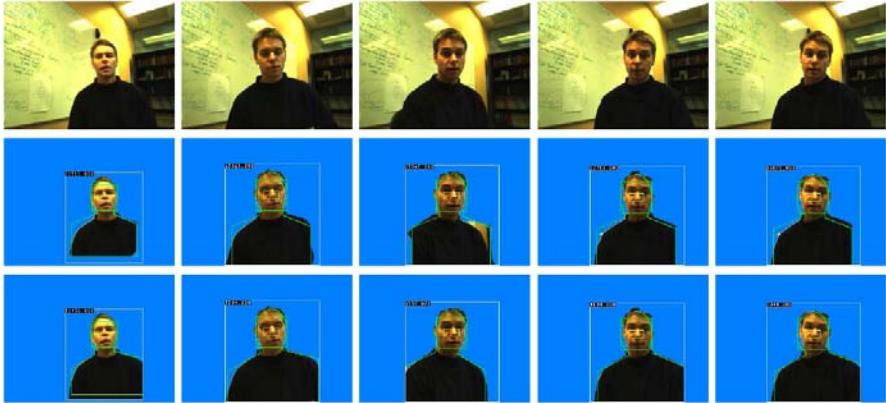
As expected, the experimental results show that the segmentation results improve considerably as we increase the amount of information in our CRF framework. Figure 3.19 shows how integrating more information in the CRF improves the segmentation results. Quantitative results for the segmentation problem are shown in Table 3.1.



**Fig. 3.19** Results showing the effect of incorporating a shape prior on the segmentation results. The first image is the original image to be segmented. The second, third and fourth images show the segmentation results obtained using colour, colour + smoothness prior and colour + smoothness + shape prior respectively.

**Table 3.1** Quantitative segmentation results. The table shows the effect of adding more information in the Bayesian framework on the quantitative segmentation accuracy. The accuracy was computed over all the pixels in the image. The ground truth for the data used in this experiment was generated by hand labelling the foreground and background regions in the images.

Information Used	Correct object pixels	All correct pixels
Colour	45.73%	95.2%
Colour + GMM	82.48%	96.9%
Colour + GMM +Shape	97.43%	99.4%



**Fig. 3.20** Segmentation results using the 2D upper body model. The first row shows some frames from the video sequence. The second row shows the initial values of the pose parameters of the model and the resulting segmentations. The last row shows the final pose estimate and segmentation obtained using our method.

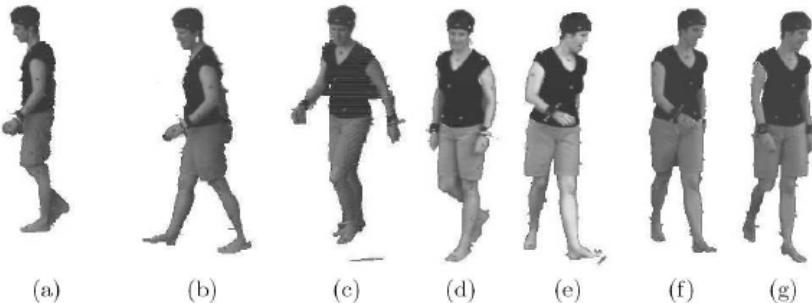
In order to demonstrate the performance of our method, we compare our segmentation results with those obtained using the method proposed in [62]. It can be seen from the results in Figure 3.21 that the segmentations obtained using the method of [62] are not accurate: They contain “speckles” and often segment the shadows of the feet as foreground. This is expected as they use only a pixelwise term to differentiate the background from the foreground and do not incorporate any spatial term which could offer a better “smoothing”. In contrast, POSECUT which uses a pairwise potential term (as any standard graph cut approach) and a shape prior (which guarantees a human-like segmentation), is able to provide accurate results.

Our experiments on segmenting humans using the 2D upper body model (Figure 3.14) also produced good results. For these experiments, video sequences from the Microsoft Research bilayer video segmentation dataset [36] were used. The results of our method are shown in Figure 3.20.

### Segmentation and Pose Estimation

Figures 3.22 and 3.23 present the segmentations and the pose estimates obtained using POSECUT. The first data set comprises of three views of human walking circularly. The time needed for computation of the 3D pose estimate, on an Intel Pentium 2GHz machine, when dealing with  $644 \times 484$  images, is about 50 seconds per view<sup>10</sup>. As shown in these Figures, the pose estimates match the original images accurately. In Figures 3.22 and 3.23 it should be noted that the appearance models of the foreground and background are quite similar: for instance, in Figure 3.23, the clothes of the subject are black in colour and the floor in the background is

<sup>10</sup> However, this could be speeded up by computing the parameters of the CRF in an FPGA (Field programmable gate array).

**Original:****Grimson :****POSECUT:**

(a)

(b)

(c)

(d)

(e)

(f)

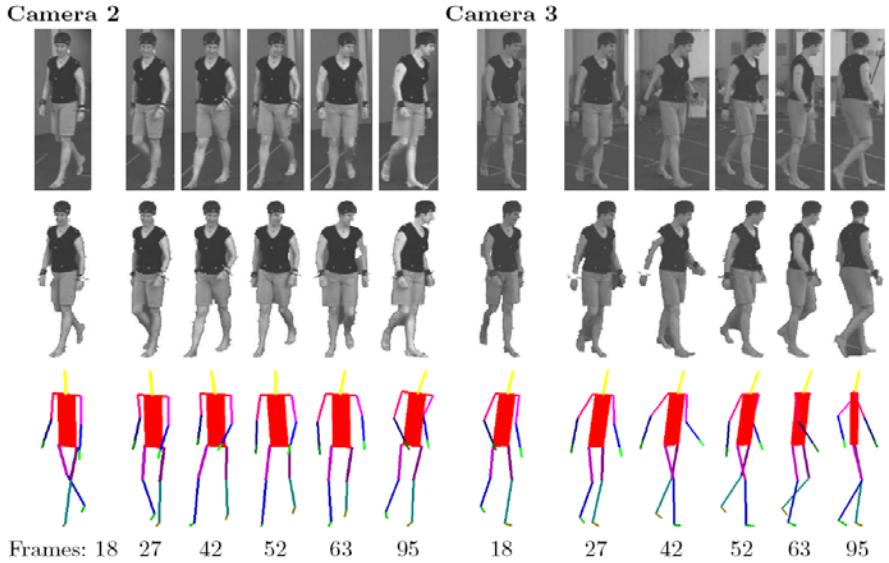
(g)

**Fig. 3.21** Segmentation results obtained by Grimson-Stauffer [62] and POSECUT.

rather dark. The accuracy of the segmentation obtained in such challenging conditions demonstrates the robustness of POSECUT. An interesting fact to observe in Figure 3.22 about frame 95 is that the torso rotation of the stickman does not exactly conform with the original pose of the object. However, the segmentation of these frames is still accurate.

### 3.7.4 Shape Priors for Reconstruction

Obtaining a 3D reconstruction of an object from multiple images is a fundamental problem in computer vision. Reflecting the importance of the problem a number of

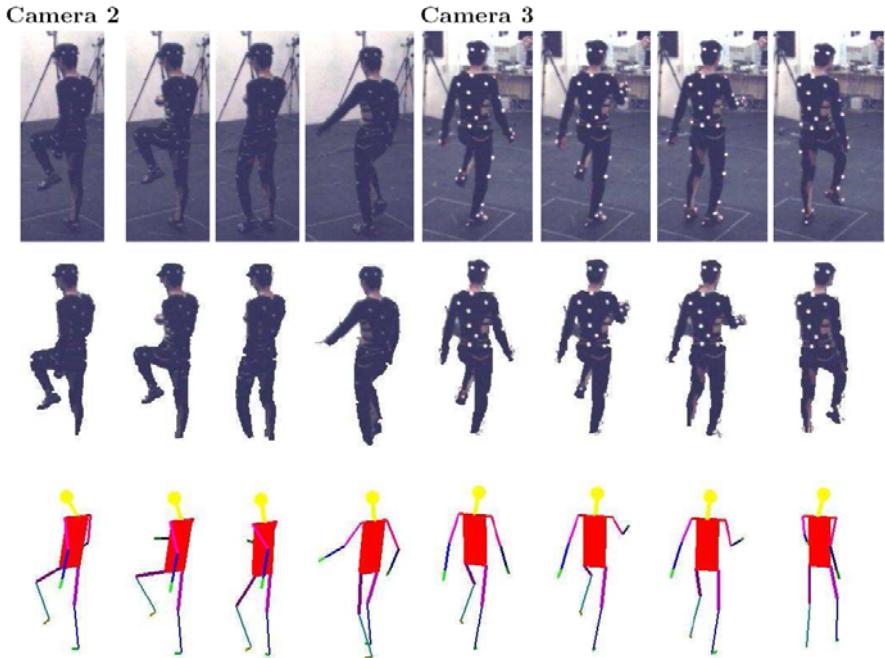


**Fig. 3.22** Segmentation (middle) and pose estimation (bottom) results from POSECUT.

methods have been proposed for its solution. These range from methods such as shape from silhouettes [65] and space carving [41] to image based methods [55]. However, the problem of obtaining accurate reconstructions from sparse multiple views still remains far from being solved. The primary problem afflicting reconstruction methods is the inherent ambiguity in the problem (as shown in Figure 3.24) which arises from the many-one nature of the mapping that relates 3D objects and their images.

Intuitively the ambiguity in the object reconstruction can be overcome by using prior knowledge. Researchers have long understood this fact and weak priors such as surface smoothness have been used in a number of methods [37, 61, 71]. Such priors help in recovering from the errors caused by noisy data. Although they improve results, they are weak and do not carry enough information to guarantee a unique solution. Taking inspiration from the success of using strong prior knowledge for image segmentation, we use 3D shape priors to overcome the ambiguity inherent in the problem of 3D reconstruction from multiple views.

Our framework uses a volumetric scene representation and integrates conventional reconstruction measures such as photoconsistency, surface smoothness and visual hull membership with a strong object specific prior. Simple parametric models of objects are used as strong priors in our framework. Our method not only gives an accurate object reconstruction, but also provides us the pose or state of the object being reconstructed. This work previously appeared in [64].

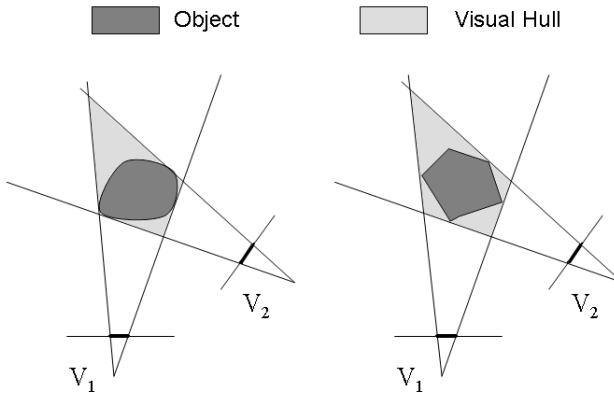


**Fig. 3.23** Segmentation (middle row) and pose estimation (bottom row) results obtained using POSECUT. Observe that although the foreground and background appearances are similar, our algorithm is able to obtain good segmentations.

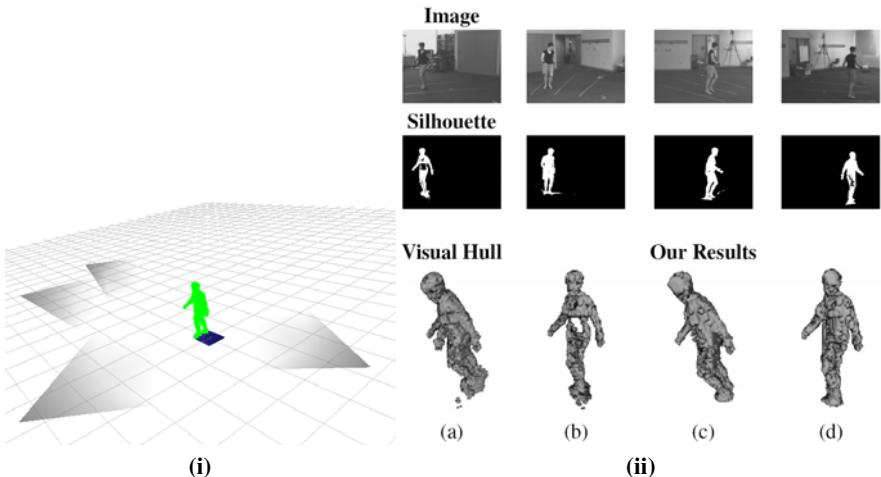
## Experimental Results

The data set for our experiments consists of video sequences of four views of a human subject walking in a circle. This data set was earlier used in [4]. It comes with silhouettes of the human subject obtained using pixel wise background intensity modelling. The positions and orientations of the 4 cameras with respect to the object are shown in Figure 3.25(i).

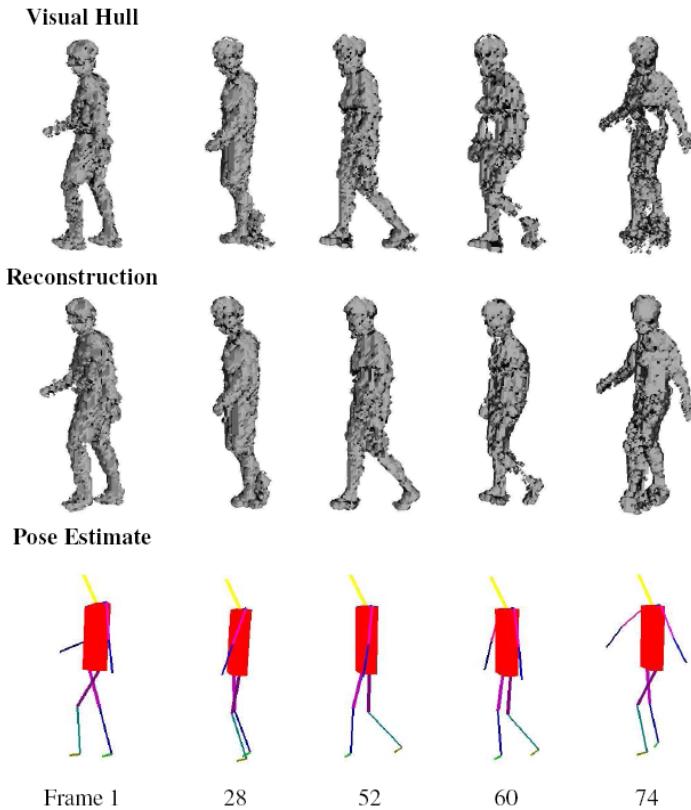
The first step in our method is the computation of the visual hull. The procedure starts with the quantization of the volume of interest as a grid of cubical voxels of equal size. Once this is done, each voxel center is projected into the input images. If any of the projections falls outside the silhouette, then the voxel is discarded. All remaining voxels constitute the visual hull. Some visual hull results are shown in Figure 3.25(ii). It can be observed that because of the skewed distribution of the cameras, the visual hull is quite different from the true object reconstruction. Further, as object segmentations are not accurate, it has large errors. The prominent defects in the visual hull results include: (i) the presence of holes because of segmentation errors in the object silhouettes (bottom row (b)), (ii) the presence of auxiliary parts caused by shadows, (iii) the *third-arm effect* resulting from self-occlusion and ambiguity in the reconstruction due to the small number of views (bottom row (a)).



**Fig. 3.24** Ambiguity in object reconstruction due to few views. The Figure shows how two completely different objects can have the same visual hull. Further, if both objects have the same colour, the photo hull and their projections on multiple viewpoints would also be the same.



**Fig. 3.25** i) Camera Positions and Reconstruction. The Figure shows the position and orientations of the four cameras which were used to obtain the images which constituted the dataset for our first experiment. We also see the reconstruction result generated by our method. ii) 3D Object Reconstruction using Strong Object-Specific priors. The first and second rows show the images and silhouettes used as the data. Two views of the visual hull generated using the data are shown in the first two columns of the bottom row ((a) and (b)). The visual hull is noisy and contains artifacts like the spurious third arm caused by the ambiguity in the problem. We are able to overcome such problems by using strong prior knowledge. The reconstructions obtained by our method are shown in column 3 and 4 ((c) and (d)).



**Fig. 3.26** Pose inference and 3D object reconstruction results. The data used in this experiment is taken from [4]. It consists of 4 views of a human subject walking in a circular path. Middle row: Reconstruction result. Bottom row: Pose estimate. Observe that we are able to get excellent reconstruction and pose estimation results even when the visual hull contains large errors (as seen in frame 60 and 74).

It can be seen that our reconstruction results do not suffer from these errors (bottom row (c) and (d)). The final results of our method for a few frames of the human walking sequence are shown in Figure 3.26.

### 3.7.5 Discussion

Localizing the object in the image and inferring its pose is a computationally expensive task. Once a rough estimate of the object pose is obtained, the segmentation can be computed extremely efficiently using graph cuts [10]. In our work on real time face detection and segmentation [52], we showed how an off the shelf face-detector such as the one described in [70] can be coupled with a CRF to get accurate segmentation and improved face detection results in real time.



**Fig. 3.27** Real time face segmentation using face detection. The first image in the first row shows the original image. The second image shows the face detection results. The third image shows the segmentation obtained using shape priors generated from the detection and localization results.

The object (face) localization estimate (obtained from any generic face detector) was incorporated in a discriminative CRF framework to obtain robust and accurate face segmentation results as shown in Figure 3.27. The energy  $E(\mathbf{x}^*)$  of any segmentation solution  $\mathbf{x}^*$  is the negative log of the probability, and can be viewed as a measure of how uncertain that solution is. The higher the energy of a segmentation, the lower the probability that it is a good segmentation. Intuitively, if the face detection is correct, the resulting segmentation obtained from our method should have high probability and hence have low energy compared to that of false detections. This characteristic of the energy of the segmentation solution can be used to prune out false face detections thus improving the face detection accuracy. The procedure is illustrated in Figure 3.28. A similar strategy was recently used in [50].

### 3.7.6 Summary and Future Work

This work sets out a novel method for performing simultaneous segmentation and 3D pose estimation (POSECUT). The problem is formulated in a Bayesian



**Fig. 3.28** Pruning false object detections. The Figure shows an image from the INRIA pedestrian data set. After running our algorithm, we obtain four face segmentations, one of which (the one bounded by a black square) is a false detection. The energy-per-pixel values obtained for the true detections were 74, 82 and 83 while that for the false detection was 87. As you can see the energy of false detection is higher than that of the true detections, and can be used to detect and remove it.

framework which has the ability to utilize all information available (prior as well as observed data) to obtain good results. We showed how a rough pose-specific shape prior could be used to improve segmentation results significantly. We also gave a new formulation of the pose inference problem as an energy minimization problem and showed how it could be efficiently solved using dynamic graph cuts. The experiments demonstrate that our method is able to obtain excellent segmentation and pose estimation results. This method was recently also used for the problem of reconstructing objects from multiple views [64].

### Searching over Pose Manifolds

It is well known that the set of all human poses constitutes a low-dimensional manifold in the complete pose space [68, 58]. Most work in exploiting this fact for human pose inference has been limited to finding linear manifolds in pose spaces. The last few years have seen the emergence of non-linear dimensionality reduction techniques for solving the pose inference problem [59]. Recently, Urtasun *et al.* [68] showed how Scaled Gaussian Process Latent Variable Models (SGPLVM) can be used to learn prior models of human pose for 3D people tracking. They showed impressive pose inference results using monocular data. Optimizing over a parametrization of this low dimensional space instead of the 26D pose vector would intuitively improve both the accuracy and computation efficiency of our algorithm. Thus the use of dimensionality reduction algorithms is an important area to be investigated. The directions for future work also include using an appearance model per limb, which being more discriminative could help provide more accurate segmentations and pose estimates.

## 3.8 Measuring Uncertainty in Graph Cut Solutions

Over the years researchers have asked the question whether it might be possible to compute a measure of uncertainty associated with the graph cut solutions. In this Section we answer this particular question positively by showing how the min-marginals associated with the label assignments of a random field can be efficiently computed using a new algorithm based on dynamic graph cuts. The min-marginal energies obtained by our proposed algorithm are exact, as opposed to the ones obtained from other inference algorithms like loopy belief propagation and generalized belief propagation. We also show how these min-marginals can be used to compute a confidence measure for label assignments in the image segmentation problem.

Graph cuts based minimization algorithms do not provide an uncertainty measure associated with the solution they produce. This is a serious drawback since researchers using these algorithms do not obtain any information regarding the probability of a particular latent variable assignment in a graph cut solution. Inference algorithms like Loopy Belief Propagation (LBP) [48], Generalized Belief Propagation (GBP) [77], and Tree-reweighted message passing (TRW) [35, 72] provide the user with marginal or min-marginal energies associated with each latent variable. However, these algorithms are not guaranteed to find the optimal solution for graphs of

arbitrary topology. Note that for tree-structured graphs, the simple max-product belief propagation algorithm gives the exact max-marginal probabilities/min-marginal energies<sup>11</sup> for different label assignments in  $O(nl^2)$  time where  $n$  is the number of latent variables, and  $l$  is the number of labels a latent variable can take.

We address the problem of efficiently computing the min-marginals associated with the label assignments of any latent variable in a MRF. Our method works on all MRFs that can be solved exactly using graph cuts. First, we give the definition of *flow potentials* (defined in Section 3.8.1) of a graph node. We show that the min-marginals associated with the labellings of a binary random variable are related to the flow-potentials of the node representing that variable in the graph constructed in the energy minimization procedure. In fact the exact min-marginal energies can be found by computing these *flow-potentials*. We then show how flow potential computation is equivalent to minimizing *projections* of the original energy function<sup>12</sup>.

Minimizing a *projection* of an energy function is a computationally expensive operation and requires a graph cut to be computed. In order to obtain the min-marginals corresponding to all label assignments of all random variables, we need to compute a graph cut  $O(nl)$  number of times. We present an algorithm based on dynamic graph cuts [33] which solves these  $O(nl)$  graph cuts extremely quickly. Our experiments show that the running time of this algorithm i.e., the time taken for it to compute the min-marginals corresponding to all latent variable label assignments is of the same order of magnitude as the time taken to solve a single st-mincut problem.

### 3.8.1 Preliminaries

As explained in Section 3.1, a pairwise MRF can be solved by minimizing a second order energy function. The energy of the MAP configuration of the MRF can be computed by solving the problem:

$$\psi(\theta) = \min_{\mathbf{x} \in \mathcal{L}} E(\mathbf{x} | \theta). \quad (3.30)$$

The MAP solution of the MRF will be referred to as the *optimal solution*.

#### Min-Marginal Energies

A min-marginal is a function that provides information about the minimum values of the energy  $E$  under different constraints. Following the notation of [35], we define the min-marginal energies  $\psi_{v;j}, \psi_{uv;ij}$  as:

---

<sup>11</sup> We will explain the relation between max-marginal probabilities and min-marginal energies later in Section 3.8.1. To make our notation consistent with recent work in graph cuts, we formulate the problem in terms of min-marginal energies (subsequently referred to as simply min-marginals).

<sup>12</sup> A projection of the function  $f(x_1, x_2, \dots, x_n)$  can be obtained by fixing the values of some of the variables in the function  $f(\cdot)$ . For instance  $f_1(x_2, \dots, x_n) = f(0, x_2, \dots, x_n)$  is a projection of the function  $f(\cdot)$ .

$$\begin{aligned}\psi_{v;j}(\theta) &= \min_{\mathbf{x} \in \mathbf{L}, x_v=j} E(\mathbf{x}|\theta), \quad \text{and} \\ \psi_{uv;ij}(\theta) &= \min_{\mathbf{x} \in \mathbf{L}, x_u=i, x_v=j} E(\mathbf{x}|\theta).\end{aligned}\tag{3.31}$$

In words, given an energy function  $E$  whose value depends on the variables  $(x_1, \dots, x_n)$ ,  $\psi_{v;j}(\theta)$  represents the minimum energy value obtained if we fix the value of variable  $x_v$  to  $j$  and minimize over all remaining variables. Similarly,  $\psi_{uv;ij}(\theta)$  represents the value of the minimum energy in the case when the values of variables  $x_u$  and  $x_v$  are fixed to  $i$  and  $j$  respectively.

### Uncertainty in Label Assignments

Now we show how min-marginals can be used to compute a confidence measure for a particular latent variable label assignment. Given the function  $p(\mathbf{x}|\mathbf{D})$ , which specifies the probability of a configuration of the MRF, the max-marginal  $\mu_{v;j}$  gives us the value of the maximum probability over all possible configurations of the MRF in which  $x_v = j$ . Formally, it is defined as:

$$\mu_{v;j} = \max_{\mathbf{x} \in \mathbf{L}, x_v=j} p(\mathbf{x}|\mathbf{D})\tag{3.32}$$

Inference algorithms like max-product belief propagation produce the max-marginals along with the MAP solution. These max-marginals can be used to obtain a confidence measure  $\sigma$  for any latent variable labelling as:

$$\sigma_{v;j} = \frac{\max_{\mathbf{x} \in \mathbf{L}, x_v=j} p(\mathbf{x}|\mathbf{D})}{\sum_{k \in \mathcal{L}} \max_{\mathbf{x} \in \mathbf{L}, x_v=k} p(\mathbf{x}|\mathbf{D})} = \frac{\mu_{v;j}}{\sum_{k \in \mathcal{L}} \mu_{v;k}}\tag{3.33}$$

where  $\sigma_{v;j}$  is the confidence for the latent variable  $x_v$  taking label  $j$ . This is the ratio of the max-marginal corresponding to the label assignment  $x_v = j$  to the sum of the max-marginals for all possible label assignments.

We now proceed to show how these max-marginals can be obtained from the min-marginal energies computed by our algorithm. Recall from equation (3.3) that the energy and probability of a labelling are related as:

$$E(\mathbf{x}) = -\log p(\mathbf{x}|\mathbf{D}) - \text{const}\tag{3.34}$$

Substituting the value of  $p(\mathbf{x}|\mathbf{D})$  from equation (3.34) in equation (3.32), we get

$$\mu_{v;j} = \max_{\mathbf{x} \in \mathbf{L}, x_v=j} (\exp(-E(\mathbf{x}|\theta)) - \text{const})\tag{3.35}$$

$$= \frac{1}{Z} \exp\left(-\min_{\mathbf{x} \in \mathcal{X}; x_v=j} E(\mathbf{x}|\theta)\right),\tag{3.36}$$

where  $Z$  is the partition function. Combining this with equation (3.31a), we get

$$\mu_{v;j} = \frac{1}{Z} \exp(-\psi_{v;j}(\theta)).\tag{3.37}$$

As an example consider a binary label object-background image segmentation problem, where there are two possible labels i.e., object ('ob') and background ('bg'). The confidence measure  $\sigma_{v;ob}$  associated with the pixel  $v$  being labelled as object can be computed as:

$$\sigma_{v;ob} = \frac{\mu_{v;ob}}{\mu_{v;ob} + \mu_{v;bg}} = \frac{\frac{1}{Z} \exp(-\psi_{v;ob}(\theta))}{\frac{1}{Z} \exp(-\psi_{v;ob}(\theta)) + \frac{1}{Z} \exp(-\psi_{v;bg}(\theta))}, \quad (3.38)$$

$$\text{or } \sigma_{v;ob} = \frac{\exp(-\psi_{v;ob}(\theta))}{\exp(-\psi_{v;ob}(\theta)) + \exp(-\psi_{v;bg}(\theta))} \quad (3.39)$$

Note that the Z's cancel and thus we can compute the confidence measure from the min-marginal energies alone without knowledge of the partition function.

## Flow Potentials in Graphs

Given a directed weighted graph  $G(V, E, C)$  with non-negative edge weights and flows  $f$  flowing through the edges  $E$ , we define the *source/sink flow potential* of a graph node  $v \in V$  as the maximum amount of net flow that can be pumped into/from it without invalidating any edge capacity (3.6) or mass balance constraint (3.7) with the exception of the mass balance constraint of the node  $v$  itself. Formally, we can define the source flow potential of node  $v$  as:

$$f_v^s = \max_{\mathbf{f}} \sum_{i \in N(v)} f_{iv} - f_{vi}$$

Subject to:

$$0 \leq f_{ij} \leq c_{ij} \quad \forall (i, j) \in E, \text{ and} \quad (3.40)$$

$$\sum_{i \in N(j) \setminus \{s, t\}} (f_{ji} - f_{ij}) = f_{sj} - f_{jt} \quad \forall j \in V \setminus \{s, t, v\} \quad (3.41)$$

where  $\max_{\mathbf{f}}$  represents the maximization over the set of all edge flows

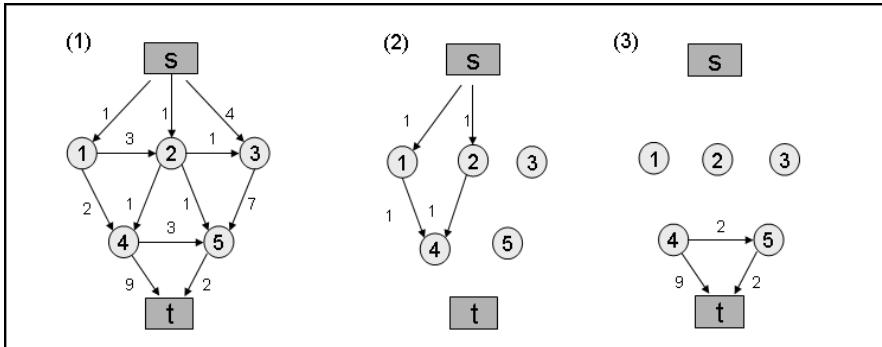
$$\mathbf{f} = \{f_{ij}, \forall (i, j) \in E\}. \quad (3.42)$$

Similarly, the *sink flow potential*  $f_v^t$  of a graph node  $v$  is defined as:

$$f_v^t = \max_{\mathbf{f}} \sum_{i \in N(v)} f_{vi} - f_{tv} \quad (3.43)$$

subject to constraints (3.40) and (3.41).

The computation of a flow potential of a node is not a trivial process and in essence requires a graph cut to be computed as explained in figure 3.30. The flow potentials of a particular graph node are shown in figure 3.29. Note that in a residual graph  $G(f_{\max})$  where  $f_{\max}$  is the maximum flow, all nodes on the sink side of the st-mincut are disconnected from the source and thus have the source flow potential



**Fig. 3.29** Flow potentials of graph nodes. The figure shows a directed graph having seven nodes, two of which are the terminal nodes, the source  $s$  and the sink  $t$ . The number associated with each directed edge in this graph is a capacity which tells us the maximum amount of flow that can be passed through it in the direction of the arrow. The flow potentials for node 4 in this graph when no flow is passing through any of the edges are  $f_4^s = 2$  and  $f_4^t = 11$ .

equal to zero. Similarly, all nodes belonging to the source have the sink flow potential equal to zero. We will later show that the flow-potentials we have just defined are intimately linked to the min-marginal energies of latent variable label assignments.

### 3.8.2 Computing Min-Marginals Using Graph Cuts

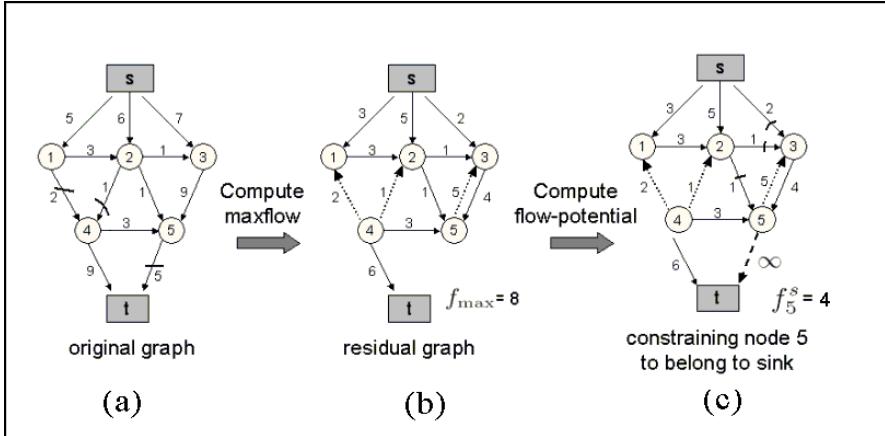
We now explain the procedure for the computation of min-marginal energies using graph cuts. The total flow  $f_{\text{total}}$  flowing from the source  $s$  to the sink  $t$  in a graph can be found by computing the difference between the total amount of flow coming in to a terminal node and that going out as:

$$f_{\text{total}} = \sum_{i \in N(s)} (f_{si} - f_{is}) = \sum_{i \in N(t)} (f_{it} - f_{ti}). \quad (3.44)$$

The cost of the st-mincut in an energy representing graph is equal to the energy of the optimal configuration. From the Ford-Fulkerson theorem, this is also equal to the maximum amount of flow  $f_{\max}$  that can be transferred from the source to the sink. Hence, from the minimum energy (3.30) and total flow equation (3.44) for a graph in which maxflow has been achieved i.e.  $f_{\text{total}} = f_{\max}$ , we obtain:

$$\psi(\theta) = \min_{\mathbf{x} \in \mathbf{L}} E(\mathbf{x} | \theta) = f_{\max} = \sum_{i \in N(s)} (f_{si} - f_{is}). \quad (3.45)$$

Note that flow cannot be pushed into the source i.e.  $f_{is} = 0, \forall i \in V$ , thus  $\psi(\theta) = \sum_{i \in N(s)} f_{si}$ . The MAP configuration  $\mathbf{x}^*$  of a MRF is the one having the least energy and is defined as  $\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbf{L}} E(\mathbf{x} | \theta)$ .



**Fig. 3.30** Computing min-marginals using graph cuts. In (a) we see the graph representing the original energy function. This is used to compute the minimum value of the energy  $\psi(\theta)$  which is equal to the max-flow  $f_{\max} = 8$ . The residual graph obtained after the computation of max-flow is shown in (b). In (c) we show how the flow-potential  $f_5^s$  can be computed in the residual graph by adding an infinite capacity edge between it and the sink and computing the max-flow again. The addition of this new edge constrains node 5 to belong to sink side of the st-cut. A max-flow computation in the graph (c) yields  $f_5^s = 4$ . This from theorem 1, we obtain the min-marginal  $\psi_{5;c} = 8 + 4 = 12$ , where  $T(c) = \text{source}(s)$ . The dotted arrows in (b) and (c) correspond to edges in the residual graph whose residual capacities are due to flow passing through the edges in their opposite direction.

Let  $a$  be the label for random variable  $x_v$  under the MAP solution and  $b$  be any label other than  $a$ . Then in the case of  $x_v = a$ , the min-marginal energy  $\psi_{v;x_v^*}(\theta)$  is equal to the minimum energy i.e.

$$E(\mathbf{x}|\theta) = \psi(\theta) \quad (3.46)$$

Thus it can be seen that the maximum flow equals the min-marginals for the case when the latent variables take their respective MAP labels.

The min-marginal energy  $\psi_{v;b}(\theta)$  corresponding to the non-optimal label  $b$  can be computed by finding the minimum value of the energy function projection  $E'$  obtained by constraining the value of  $x_v$  to  $b$  as:

$$\psi_{v;b}(\theta) = \min_{\mathbf{x} \in \mathcal{L}^n, x_v = b} E(\mathbf{x}|\theta) \quad (3.47)$$

$$\text{or, } \psi_{v;b}(\theta) = \min_{(\mathbf{x}-x_v) \in \mathcal{L}^{n-1}} E(x_1, \dots, x_{v-1}, b, x_{v+1} \dots x_n | \theta). \quad (3.48)$$

In the next sub-section, we will show that this constraint can be enforced in the original graph construction used for minimizing  $E(\mathbf{x}|\theta)$  by modifying certain edge weights ensuring that the latent variable  $x_v$  takes the label  $b$ . The exact modifications

needed in the graph for the binary label case are given first, while those required in the graph for the multi-label case are discussed later.

### 3.8.3 Min-Marginals and Flow Potentials

We now show how in the case of binary variables, flow-potentials in the residual graph  $G(f_{\max})$  are related to the min-marginal energy values. Again,  $a$  and  $b$  are used to represent the MAP and non-MAP label respectively.

**Theorem 3.1.** *The min-marginal energies of a binary latent variable  $x_v$  are equal to the sum of the max-flow and the source/sink flow potentials of the node representing it in the residual graph  $G(f_{\max})$  corresponding to the max-flow solution i.e.*

$$\psi_{v;j}(\theta) = \min_{x \in \mathbf{L}, x_v=j} E(\mathbf{x}|\theta) = \psi(\theta) + f_v^{T(j)} = f_{\max} + f_v^{T(j)} \quad (3.49)$$

where  $T(j)$  is the terminal corresponding to the label  $j$ , and  $f_{\max}$  is the value of the maximum flow in the graph  $G$  representing the energy function  $E(\mathbf{x}|\theta)$ .

**Proof.** The proof is trivial for the case where the latent variable takes the optimal label. We already know that the value of the min-marginal  $\psi_{v;a}(\theta)$  is equal to the lowest energy  $\psi(\theta)$ . Further, the flow potential of the node for the terminal corresponding to the label assignment is zero since the node is disconnected from the terminal  $T(a)$  by the minimum cut<sup>13</sup>.

We already know from (3.48) that the min-marginal  $\psi_{v;b}(\theta)$  corresponding to the non-optimal label  $b$  can be computed by finding the minimum value of the function  $E$  under the constraint  $x_v = b$ . This constraint can be enforced in our original graph (used for minimizing  $E(x|\theta)$ ) by adding an edge with infinite weight between the graph node and the terminal corresponding to the label  $a$ , and then computing the st-mincut on this updated graph<sup>14</sup>. In Section 3.8.5 we shall explain how to solve the new st-mincut problem efficiently using the dynamic graph cut algorithm proposed in the previous Section.

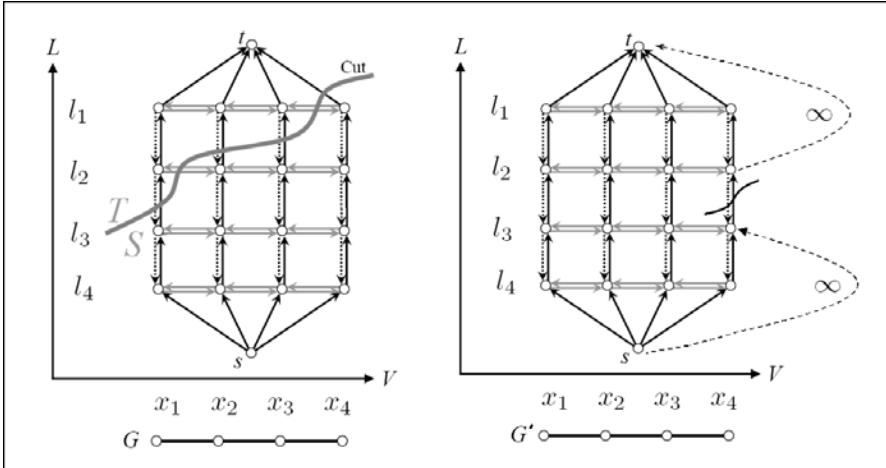
It can be easily seen that the additional amount of flow that would now flow from the source to the sink is equal to the flow potential  $f_v^{T(b)}$  of the node. Thus the value of the max-flow now becomes equal to  $\psi(\theta) + f_v^{T(b)}$  where  $T(b)$  is the terminal corresponding to the label  $b$ . The whole process is shown graphically in figure 3.30.

We have shown how minimizing an energy function with constraints on the value of a latent variable, is equivalent to computing the flow potentials of a node in the

---

<sup>13</sup> The amount of flow that can be transferred from the node to the terminal  $T(a)$  in the residual graph is zero since otherwise it would contradict our assumption that the max-flow solution has been achieved.

<sup>14</sup> Adding an infinite weight edge between the node and the terminal  $T(a)$  is equivalent to putting a hard constraint on the variable  $x_v$  to have the label  $b$ . Observe that the addition of an infinite weight edge can be realized by using an edge whose weight is more than the sum of all other edges incident on the node. This condition would make sure that the edge is not saturated during the max-flow computation.



**Fig. 3.31** Graph construction for projections of energy functions involving multiple labels. The first graph  $G$  shows the graph construction proposed by Ishikawa [27] for minimizing energy functions representing MRFs involving latent variables which can take more than 2 labels. All the label sets  $\mathcal{L}, v \in V$  consist of 4 labels namely  $l_1, l_2, l_3$  and  $l_4$ . The MAP configuration of the MRF induced by the st-mincut is found by observing which data edges are cut (data edges are depicted as black arrows). Four of them are in the cut here (as seen in graph  $G$ ), representing the assignments  $x_1 = l_2, x_2 = l_3, x_3 = l_3$ , and  $x_4 = l_4$ . The graph  $G'$  representing the projection  $E' = E(x_1, x_2, x_3, l_2)$  can be obtained by inserting infinite capacity edges from the source and the sink to the tail and head node respectively of the edge representing the label  $l_2$  for latent variable  $x_4$ .

residual graph  $G(f_{\max})$ . Note that a similar procedure can be used to compute the min-marginal  $\psi_{uv;ij}(\theta)$  by taking the projection and enforcing hard constraints on pairs of latent variables.

### 3.8.4 Extension to Multiple Labels

Graph cuts can also be used to exactly optimize convex energy functions which involve variables taking multiple labels [27, 56]. Graphs representing the projections of such energy functions can be obtained by incorporating hard constraints in a fashion analogous to the one used for binary variables. In the graph construction for multiple labels proposed by Ishikawa [27], the label of a discrete latent variable is found by observing which data edge is cut. The value of a variable can be constrained or ‘fixed’ in this graph construction by making sure that the data edge corresponding to the particular label is cut. This can be realized by adding edges of infinite capacity from the source and the sink to the tail and head node of the edge respectively as shown in figure 3.31. The cost of the st-mincut in this modified graph will give the exact value of min-marginal energy associated with that particular labelling. It should be noted here that the method of Ishikawa [27] ap-

**Table 3.2** Algorithm for computing min-marginal energies using dynamic graph cuts.

1. Construct graph  $G$  for minimizing the MRF energy  $E$ .
2. Compute the maximum s-t flow in the graph. This induces the residual graph  $G_r$  consisting of unsaturated edges.
3. For computing each min-marginal, perform the following operations:
  - a. Obtain the energy projection  $E'$  corresponding to the latent variable assignment.
  - b. Construct the graph  $G'$  to minimize  $E'$ .
  - c. Use dynamic graph cut updates as given in [33] to make  $G_r$  consistent with  $G'$ , thus obtaining the new graph  $G'_r$ .
  - d. Compute the min-marginal by minimizing  $E'$  using the dynamic (optimized) st-mincut algorithm on  $G'_r$ .

plies to a restricted class of energy functions. These do not include energies with non-convex priors (such at the Potts model) which are used in many computer vision applications. Measuring uncertainty in solutions of such energies is thus still an open problem.

### 3.8.5 Minimizing Energy Function Projections Using Dynamic Graph Cuts

Having shown how min-marginals can be computed using graph cuts, we now explain how this can be done efficiently. As explained in the proof of Theorem 1, we can compute min-marginals by minimizing projections of the energy function. However, it might be thought that such a process is extremely computationally expensive as a graph cut has to be computed for every min-marginal computation. However, when modifying the graph in order to minimize the projection  $E'$  of the energy function, only a few edge weights have to be changed<sup>15</sup> as seen in figure 3.30, where only one infinite capacity edge had to be inserted in the graph. We have shown earlier how the st-mincut can be recomputed rapidly for such minimal changes in the problem by using dynamic graph cuts. Our proposed algorithm for min-marginal computation is given in Table 3.2.

### 3.8.6 Computational Complexity and Experimental Evaluation

We now discuss the computational complexity of our algorithm, and report the time taken by it to compute min-marginals in MRFs of different sizes. In step (3.4) of the algorithm given in Table 3.2, the amount of flow computed is equal to the difference in the min-marginal  $\psi_{v,j}(\theta)$  of the particular label assignment and the minimum

<sup>15</sup> The exact number of edge weights that have to be changed is of the order of the number of variables whose value is being fixed for obtaining the projection.

energy  $\psi(\theta)$ . Let  $\mathcal{Q}$  be the set of all label assignments whose corresponding min-marginals have to be computed. Then the number of augmenting paths to be found during the whole algorithm is bounded from above by:

$$U = \psi(\theta) + \sum_{q \in \mathcal{Q}} (\psi_q(\theta) - \psi(\theta)). \quad (3.50)$$

For the case of binary random variables, assuming that we want to compute all latent variable min-marginals i.e.

$$\mathcal{Q} = \{(u; i) : u \in V, i \in \mathcal{L}\}, \text{ and} \quad (3.51)$$

$$q_{max} = \max_{q \in \mathcal{Q}} (\psi_q(\theta) - \psi(\theta)), \quad (3.52)$$

the complexity of the above algorithm becomes  $O((\psi(\theta) + nq_{max})T(n, m))$ , where  $T(n, m)$  is the complexity of finding an augmenting path in the graph with  $n$  nodes and  $m$  edges and pushing flow through it. Although the worst case complexity  $T(n, m)$  of the augmentation operation is  $O(m)$ , we observe experimentally that using the dual search tree algorithm of [8], we can get a much better amortized time performance. The average time taken by our algorithm for computing the min-marginals in random MRFs of different sizes is given in table 3.3.

**Table 3.3** Time taken for min-marginal computation. For a sequence of randomly generated MRFs of a particular size and neighbourhood system, a pair of times (in seconds) is given in each cell of the table. On the left is the average time taken to compute the MAP solution using a single graph cut while on the right is the average time taken to compute the min-marginals corresponding to all latent variable label assignments. The dynamic algorithm with tree-recycling was used for this experiment. All experiments were performed on an Intel Pentium 2GHz machine.

MRF size	$10^5$	$2 \times 10^5$	$4 \times 10^5$	$8 \times 10^5$
4-neighbourhood	0.18, 0.70	0.46, 1.34	0.92, 3.156	2.17, 8.21
8-neighbourhood	0.40, 1.53	1.39, 3.59	2.42, 8.50	5.12, 15.61

### 3.8.7 Applications of Min-Marginals

Min-marginal energies have been used for a number of different purposes. However, prior to our work, the use of min-marginals in computer vision was severely restricted. This was primarily due to the fact that they were computationally expensive to compute for MRFs having a large number of latent variables. Our new algorithm is able to handle large MRFs which opens up possibilities for many new applications. For instance, in the experiments shown in figure 3.32, the time taken for computing all min-marginals for a MRF consisting of  $2 \times 10^5$  binary latent variables was 1.2 seconds. This is roughly four times the time taken for computing the MAP solution of the same MRF by solving a single st-mincut problem.



**Fig. 3.32** Image segmentation with max-marginal probabilities. The first image is a frame of the movie *Run Lola Run*. The second shows the binary foreground-background segmentation where the aim was to segment out the human. The third and fourth images shows the confidence values obtained by our algorithm for assigning pixels to be foreground and background respectively. In the image, the max-marginal probability is represented in terms of decreasing intensity of the pixel. Our algorithm took 1.2 seconds for computing the max-marginal probabilities for each latent variable label assignment. The time taken to compute the MAP solution was 0.3 seconds.

### Min-Marginals as a Confidence Measure

We had shown in Section 3.8.1 how min-marginals can be used to compute a confidence measure for any latent variable assignment in a MRF. Figure 3.32 shows the confidence values obtained for the MRF used for modeling the two label (foreground and background) image-segmentation problem as defined in [7]. Ideally we would like the confidence map to be black and white showing extremely ‘low’ or ‘high’ confidence for a particular label assignment. However, as can be seen from the result, the confidence map contains regions of different shades of grey. Such confidence maps can be used for many vision applications. For instance, they could be used in interactive image segmentation to direct user interaction at regions which have high uncertainty. They can also be applied in coarse-to-fine techniques for efficient computation of low level vision problems. Here confidence maps could be used to isolate variables which have low confidence in the optimal label assignment. These variables can be solved at higher resolution to get a better solution.

### Computing the $M$ most Probable Configurations

One of the most important uses of min-marginals has been to find the  $M$  most probable configurations (or labellings) for latent variables in a Bayesian network [76].

Dawid [14] showed how min-marginals on junction trees can be computed. This method was later used by [47] to find the M most probable configurations of a probabilistic graphical network. The method of [14] is guaranteed to run in polynomial time for tree-structured networks. However, for arbitrary graphs, its worst case complexity is exponential in the number of the nodes in the graphical model. By using our method, the M most probable solutions of some graphical models with loops can be computed in reasonable time.

We end this Section by giving a brief summary. We have addressed the long-standing problem of computing the exact min-marginals for graphs with arbitrary topology in polynomial time. We propose a novel algorithm based on dynamic graph cuts [33] that computes the min-marginals extremely efficiently. Our algorithm makes it feasible to compute exact min-marginals for MRFs with large number of latent variables. This opens up many new applications for min-marginals which were not feasible earlier.

## References

1. Agarwal, A., Triggs, B.: 3d human pose from silhouettes by relevance vector regression. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 882–888 (2004)
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice Hall, Eaglewood Cliffs (1993)
3. Alahari, K., Kohli, P., Torr, P.H.S.: Reduce, reuse & recycle: Efficiently solving multi-label mrf. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2008)
4. Bhatia, S., Sigal, L., Isard, M., Black, M.J.: 3d human limb detection using space carving and multi-view eigen models. In: Proceedings of the ANM Workshop (2004)
5. Blake, A., Rother, C., Brown, M., Perez, P., Torr, P.H.S.: Interactive image segmentation using an adaptive GMMRF model. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3021, pp. 428–441. Springer, Heidelberg (2004)
6. Boros, E., Hammer, P.L.: Pseudo-boolean optimization. Discrete Applied Mathematics 123(1-3), 155–225 (2002)
7. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In: Proceedings of the International Conference on Computer Vision, pp. 105–112 (2001)
8. Boykov, Y., Kolmogorov, V.: An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. IEEE Transaction on Pattern Analysis and Machine Intelligence 26(9), 1124–1137 (2004)
9. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transaction on Pattern Analysis and Machine Intelligence 23(11), 1222–1239 (2001)
10. Bray, M., Kohli, P., Torr, P.H.S.: Posecut: Simultaneous segmentation and 3d pose estimation of humans using dynamic graph-cuts. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 642–655. Springer, Heidelberg (2006)
11. Chiang, Y., Tamassia, R.: Dynamic algorithms in computational geometry. IEEE Special Issue on Computational Geometry 80, 362–381 (1992)
12. Cohen, R.F., Tamassia, R.: Dynamic expression trees and their applications. In: Proceedings of the Symposium on Discrete Algorithms, pp. 52–61 (1991)

13. Cremers, D., Osher, S., Soatto, S.: Kernel density estimation and intrinsic alignment for shape priors in level set segmentation. *International Journal of Computer Vision* 69(3), 335–351 (2006)
14. Dawid, P.: Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing* 2, 25–36 (1992)
15. Deutscher, J., Davison, A., Reid, I.: Automatic partitioning of high dimensional search spaces associated with articulated body motion capture. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 669–676 (2001)
16. Dinic, E.A.: Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Math. Dokl.* 11, 1277–1280 (1970)
17. Felzenszwalb, P.F., Huttenlocher, D.: Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell University (2004)
18. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient matching of pictorial structures. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 2066–2073 (2000)
19. Flach, B.: Strukturelle bilderkennung. Technical report, Universit at Dresden (2002)
20. Ford, L.R., Fulkerson, D.R.: *Flows in Networks*. Princeton University Press, Princeton (1962)
21. Freedman, D., Zhang, T.: Interactive graph cut based segmentation with shape priors. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 755–762 (2005)
22. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing* 18:18, 30–55 (1989)
23. Gavrila, D.M., Davis, L.S.: 3D model-based tracking of humans in action: a multi-view approach. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 73–80 (1996)
24. Greig, D., Porteous, B., Seheult, A.: Exact maximum a posteriori estimation for binary images. *RoyalStat* 51(2), 271–279 (1989)
25. Hengel, A., Dick, A., Thormhlen, T., Ward, B., Torr, P.H.S.: Rapid interactive modelling from video with graph cuts. In: *Proceedings of Eurographics* (2006)
26. Huang, R., Pavlovic, V., Metaxas, D.N.: A graphical model framework for coupling MRFs and deformable models. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 739–746 (2004)
27. Ishikawa, H.: Exact optimization for markov random fields with convex priors. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 25, 1333–1336 (2003)
28. Ishikawa, H., Geiger, D.: Occlusions, discontinuities, and epipolar lines in stereo. In: Burkhardt, H.-J., Neumann, B. (eds.) *ECCV 1998. LNCS*, vol. 1406, pp. 232–248. Springer, Heidelberg (1998)
29. Ishikawa, H., Geiger, D.: Segmentation by grouping junctions. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 125–131 (1998)
30. Juan, O., Boykov, Y.: Active graph cuts. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 1023–1029 (2006)
31. Kehl, R., Bray, M., Van Gool, L.: Full body tracking from multiple views using stochastic sampling. In: *Proceedings of the International conference on Computer Vision and Pattern Recognition*, pp. 129–136 (2005)
32. Kohli, P., Kumar, M.P., Torr, P.H.S.:  $P^3$  and beyond: Solving energies with higher order cliques. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2007)

33. Kohli, P., Torr, P.H.S.: Efficiently solving dynamic markov random fields using graph cuts. In: Proceedings of the International Conference on Computer Vision, pp. 922–929 (2005)
34. Kohli, P., Torr, P.H.S.: Measuring uncertainty in graph cut solutions: Efficiently computing min-marginal energies using dynamic graph cuts. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 30–43. Springer, Heidelberg (2006)
35. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. IEEE Transaction on Pattern Analysis and Machine Intelligence 28(10), 1568–1583 (2006)
36. Kolmogorov, V., Criminisi, A., Blake, A., Cross, G., Rother, C.: Bi-layer segmentation of binocular stereo video. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 407–414 (2005)
37. Kolmogorov, V., Zabih, R.: Multi-camera scene reconstruction via graph cuts. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2352, pp. 82–96. Springer, Heidelberg (2002)
38. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? IEEE Transaction on Pattern Analysis and Machine Intelligence 26(2), 147–159 (2004)
39. Komodakis, N.: A new framework for approximate labeling via graph cuts. In: Proceedings of the International Conference on Computer Vision, pp. 1018–1025 (2005)
40. Kumar, M.P., Torr, P.H.S., Zisserman, A.: Obj cut. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 18–25 (2005)
41. Kutulakos, K.N., Seitz, M.: A theory of shape by space carving. International Journal of Computer Vision 38(3) (2000)
42. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In: Proceedings of the International Conference on Machine Learning, pp. 282–289 (2001)
43. Lan, X., Huttenlocher, D.P.: Beyond trees: Common-factor models for 2d human pose recovery. In: Proceedings of the International Conference on Computer Vision, pp. 470–477 (2005)
44. Lempitsky, V.S., Roth, S., Rother, C.: Fusionflow: Discrete-continuous optimization for optical flow estimation. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2008)
45. Leventon, M.E., Grimson, W.E.L., Faugeras, O.D.: Statistical shape influence in geodesic active contours. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 1316–1323 (2000)
46. Mori, G., Ren, X., Efros, A.A., Malik, J.: Recovering human body configurations: Combining segmentation and recognition. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 326–333 (2004)
47. Nilsson, D.: An efficient algorithm for finding the m most probable configurations in bayesian networks. Statistics and Computing 8(2), 159–173 (1998)
48. Pearl, J.: Fusion, propagation, and structuring in belief networks. Artificial Intelligence 29(3), 241–288 (1986)
49. Press, W., Flannery, B., Teukolsky, S., Vetterling, W.: Numerical recipes in C. Cambridge Uni. Press, Cambridge (1988)
50. Ramanan, D.: Using segmentation to verify object hypotheses. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
51. Ramanan, D., Forsyth, D.A.: Finding and tracking people from the bottom up. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 467–474 (2003)

52. Rihan, J., Kohli, P., Torr, P.H.S.: Objcut for face detection. In: Kalra, P.K., Peleg, S. (eds.) ICVGIP 2006. LNCS, vol. 4338, pp. 576–584. Springer, Heidelberg (2006)
53. Roth, S., Black, M.J.: Fields of experts: A framework for learning image priors. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 860–867 (2005)
54. Rother, C., Kolmogorov, V., Lempitsky, V., Szummer, M.: Optimizing binary MRFs via extended roof duality. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
55. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision 47(1-3), 7–42 (2002)
56. Schlesinger, D., Flach, B.: Transforming an arbitrary minsum problem into a binary one. Technical Report TUD-FI06-01, Dresden University of Technology (2006)
57. Shakhnarovich, G., Viola, P., Darrell, T.J.: Fast pose estimation with parameter-sensitive hashing. In: Proceedings of the International Conference on Computer Vision, pp. 750–757 (2003)
58. Sidenbladh, H., Black, M.J., Fleet, D.J.: Stochastic tracking of 3D human figures using 2D image motion. In: Vernon, D. (ed.) ECCV 2000. LNCS, vol. 1843, pp. 702–718. Springer, Heidelberg (2000)
59. Sminchisescu, C., Jepson, A.D.: Generative modeling for continuous non-linearly embedded visual inference. In: Proceedings of the International Conference on Machine Learning (2004)
60. Sminchisescu, C., Triggs, B.: Covariance scaled sampling for monocular 3D body tracking. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 447–454 (2001)
61. Snow, D., Viola, P., Zabih, R.: Exact voxel occupancy with graph cuts. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 345–352 (2000)
62. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 246–252 (1999)
63. Stenger, B., Thayananthan, A., Torr, P.H.S., Cipolla, R.: Filtering using a tree-based estimator. In: Proceedings of the International Conference on Computer Vision, pp. 1063–1070 (2003)
64. Sun, Y., Kohli, P., Bray, M., Torr, P.H.S.: Using strong shape priors for stereo. In: Kalra, P.K., Peleg, S. (eds.) ICVGIP 2006. LNCS, vol. 4338, pp. 882–893. Springer, Heidelberg (2006)
65. Szeliski, R.: Rapid octree construction from image sequences. Computer Vision Graphics and Image Processing 58, 23–32 (1993)
66. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M.F., Rother, C.: A comparative study of energy minimization methods for markov random fields. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 16–29. Springer, Heidelberg (2006)
67. Thorup, M.: Fully-dynamic min-cut. In: Proceedings of the ACM Symposium on Theory of Computing, pp. 224–230 (2001)
68. Urtasun, R., Fleet, D.J., Hertzmann, A., Fua, P.: Priors for people tracking from small training sets. In: Proceedings of the International Conference on Computer Vision, pp. 403–410 (2005)
69. Veksler, O.: Graph cut based optimization for MRFs with truncated convex priors. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)

70. Viola, P.A., Jones, M.J.: Robust real-time face detection. *International Journal of Computer Vision* 57(2), 137–154 (2004)
71. Vogiatzis, G., Torr, P.H.S., Cipolla, R.: Multi-view stereo via volumetric graph-cuts. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 391–398 (2005)
72. Wainwright, M.J., Jaakkola, T., Willsky, A.S.: Map estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory* 51(11), 3697–3717 (2005)
73. Wang, J., Bhat, P., Colburn, A., Agrawala, M., Cohen, M.F.: Interactive video cutout. *ACM Transaction on Graphics* 24(3), 585–594 (2005)
74. Woodford, O.J., Torr, P.H.S., Reid, I.D., Fitzgibbon, A.W.: Global stereo reconstruction under second order smoothness priors. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2008)
75. Xiao, J., Shah, M.: Motion layer extraction in the presence of occlusion using graph cut. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 972–979 (2004)
76. Yanover, C., Weiss, Y.: Finding the m most probable configurations in arbitrary graphical models. In: *Proceedings of the International Conference on Neural Information Processing Systems* (2004)
77. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Generalized belief propagation. In: *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 689–695 (2000)
78. Zhao L., Davis, L. S.: Closely coupled object detection and segmentation. In: *Proceedings of the International Conference on Computer Vision*, pp. 454–461 (2005)

# Chapter 4

## Discriminative Graphical Models for Context-Based Classification

Sanjiv Kumar

**Abstract.** Natural image data shows significant dependencies that should be modeled appropriately to achieve good classification. Such dependencies are commonly referred to as *context* in Vision. This chapter describes Conditional Random Fields (CRFs) based discriminative models for incorporating context in a principled manner. Unlike the traditional generative Markov Random Fields (MRFs), CRFs allow the use of arbitrarily complex dependencies in the observed data along with data-dependent interactions in labels. Fast and robust parameter learning techniques for such models are described. The extensions of the standard binary CRFs to handle problems with multiclass labels or hierarchical context are also discussed. Finally, application of CRFs on contextual object detection, scene segmentation and texture recognition tasks is demonstrated.

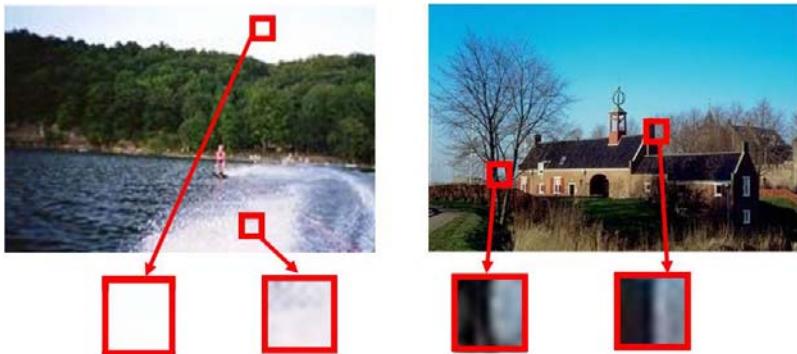
### 4.1 Contextual Dependencies in Images

One of the fundamental problems in computer vision is that of *image understanding* or *semantic scene interpretation* i.e., to interpret the scene contained in an image as a collection of meaningful entities. This may involve parsing information in the scene at different levels. Here, we focus on the problem of classification or labeling of various components in natural images, where a component may be an image pixel, a region, an object or the entire image itself.

The problem of detecting and classifying regions and objects in images is a challenging task due to ambiguities in the appearance of the visual data. The use of context can help alleviate this problem significantly. For example, as shown in Figure 4.1, just on the basis of appearance, it may be difficult to differentiate a sky patch from a water patch but their relative spatial configuration with respect to other regions removes this ambiguity. Similarly, a patch from a tree may appear locally very

---

Sanjiv Kumar  
Google Research, New York, USA  
e-mail: [sanjivk@google.com](mailto:sanjivk@google.com)



**Fig. 4.1** Classification of image components is difficult due to ambiguities in their appearance. In the left image, sky and water regions look similar while in the right image, tree and building regions look similar. Context can help resolve these ambiguities.

similar to another patch from a building (Figure 4.1, right image). But if we look at larger neighborhoods of the patch, it is easy to classify which patch is a building patch.

It is well known that natural images are not a random collection of independent pixels. The spatial arrangement of pixels (or blocks) in images is crucial to make a meaningful image. It is important to use contextual information in the form of spatial dependencies for robust analysis of images. Since these dependencies can be short-range or long-range, one would like to have total freedom in modeling data interactions without restricting oneself to small local neighborhoods. This idea forms the core of the work described in this chapter. The spatial dependencies may vary from being local to global and the challenge is how to maintain global spatial consistency using models that only need to consider relatively local dependencies.

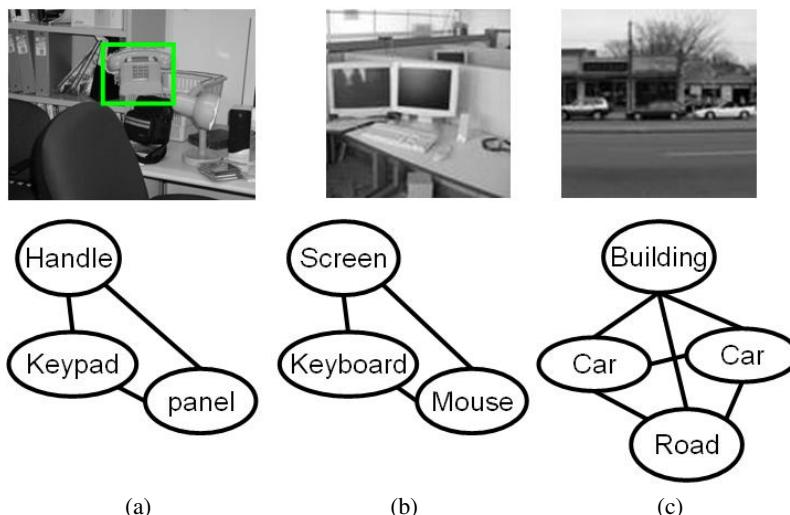
#### 4.1.1 *The Nature of Contextual Interactions*

There are several types of contextual interactions one would like to model to achieve robust classification in images. The simplest type of interaction is based on the notion of spatial smoothness of labels in natural images. According to this, neighboring pixels tend to have similar labels (except at the discontinuities). For example, if a pixel in left image in Figure 4.1 has label *sky*, there is a high probability that the neighboring pixels also have the same label except at the boundaries. In fact, the underlying smoothness of natural images forms the basis for recovering the true image from its noisy version in image denoising applications. These type of interactions are generally restricted to the pixel level. However, in addition to these, there exist significant interactions among bigger regions in images. In the previous example (Figure 4.1, left image), different semantic regions follow plausible spatial configurations (e.g., sky tends to occur above water or vegetation).

In addition to the interaction in labels, there are also complex interactions in the observed data that might be required for classification purposes. Consider the task of detecting structured textures (e.g., man-made structures such as buildings) in a given image. The data belonging to this type of textures is highly dependent on its neighbors. This is because, in man-made structures, the lines or edges at spatially adjoining regions follow some underlying organization rules rather than being random (see Figure 4.1 right image).

Now, considering the case of parts-based object detection, one would like to detect different parts of an object to form a hypothesis about the presence of the whole object. For example, in Figure 4.2 (a), we are interested in detecting a *phone*. Different parts of the phone such as handle, keypad and front panel are related to each other through geometric and, possibly, photometric constraints. The phone can be detected in the scene if we can find the locations of these parts. However, to reliably detect these parts, we need to encode not only the appearance of each individual part but also the spatial relationships among the parts. Thus, in this case, context is applied using the mutual relationships of different parts.

Finally, the contextual interactions for object detection are not limited to the parts of a single object. These may include interactions among various objects or regions in the scene. For example, as shown in Figure 4.2 (b), the presence of a monitor screen increases the probability of having a keyboard or mouse nearby. Exploiting such contextual information is crucial especially for detecting those objects



**Fig. 4.2** Context is important for the detection of objects in their natural surroundings. (a) Different parts of an object (phone) are related through geometric constraints that can help in robust detection of individual parts. (b) Different objects (monitor, keyboard and mouse) in a scene occur in restricted configurations which can help in detecting objects with impoverished appearance (e.g., mouse). (c) Context from other regions (e.g., buildings and roads) can be helpful in detecting cars).

that have impoverished appearances such as the mouse in this case. Similarly, the presence of regions such as buildings and roads in a scene restricts the possible locations a car can take in the image (Figure 4.2(c)).

To summarize, context in images can be broadly divided into two categories. First, *local context* e.g., local smoothness of pixel labels in images or interactions among different parts of an object, and second, *global context* such as interaction among bigger objects and regions in images. The challenge is how to model different types of context, which may include complex dependencies in the observed image data as well as the labels, in a principled manner. Ideally, one would like to find a computational model that can learn all relevant types of context automatically in a single consistent framework using the training data. Discriminative graphical models provide a solid platform to achieve that. Such models are by nature non-causal and are typically represented by undirected graphs. Let us first briefly review an undirected probabilistic graphical model commonly used in Computer Vision.

## 4.2 Markov Random Field (MRF)

Markov Random Fields (MRFs) are the most popular undirected graphical models in vision, which allow one to incorporate local contextual constraints in labeling problems in a principled manner. MRFs were made popular in vision by early work of Geman and Geman [4], and Besag [1]. MRFs are generally used in a probabilistic generative framework that models the joint probability of the observed data and the corresponding labels. In other words, let  $y$  be the observed data from an input image, where  $y = \{y_i\}_{i \in S}$ ,  $y_i$  is the data from the  $i^{th}$  site, and  $S$  is the set of sites. Let the corresponding labels at the image sites be given by  $x = \{x_i\}_{i \in S}$ . In the MRF framework, the posterior over the labels given the data is expressed using the Bayes' rule as,

$$P(x|y) \propto p(x,y) = P(x)p(y|x)$$

where the prior over labels,  $P(x)$  is modeled as a MRF. For computational tractability, the observation or likelihood model,  $p(y|x)$  is assumed to have a factorized form, i.e.,  $p(y|x) = \prod_{i \in S} p(y_i|x_i)$ . However, this assumption is too restrictive for several natural image analysis applications. For example, consider a class that contains man-made structures (e.g., buildings). The data belonging to such a class is highly dependent on its neighbors. This is because, in man-made structures, the lines or edges at spatially adjoining sites follow some underlying organization rules rather than being random. This is also true for a large number of texture classes that are made of structured patterns.

Another thing to note is that the interaction among labels in MRFs is modeled by the term  $P(x)$ , which is seen as a prior in the Bayesian view. The main drawback of this view is that the label interactions do not depend on the observed data  $y$ . This prohibits one from modeling data-dependent interactions in labels that are necessary for a variety of tasks. For example, while implementing local smoothness of labels in image segmentation, it may be desirable to use observed data to modulate the smoothness according to the image intensity gradients. Further, in parts based

object detection, to model interactions among object parts, we need observed data to enforce geometric (and possibly photometric) constraints. This is also the case for modeling higher level interactions between objects or regions in an image. As we will see later, discriminative graphical models allow interactions among labels based on unrestricted use of observations as necessary. This step is crucial to develop models that can incorporate interactions of different types within the same framework.

In MRF formulation of binary classification problems, the label interaction field  $P(x)$  is commonly assumed to be a homogeneous and isotropic Ising model (or Potts model for multiclass labeling problems) with only pairwise nonzero potentials. If the data likelihood  $p(y|x)$  is approximated by assuming that the observed data is conditionally independent given the labels, the posterior distribution<sup>1</sup> over labels can be written as,

$$P(x|y) = \frac{1}{Z_m} \exp\left(\sum_{i \in S} \log p(s_i(y_i)|x_i) + \sum_{i \in S} \sum_{j \in N_i} \beta_m x_i x_j\right), \quad (4.1)$$

where  $\beta_m$  is the interaction parameter of the MRF, and  $s_i(y_i)$  is a *single-site* feature vector, which uses data only from a single site  $i$ , i.e.,  $y_i$ . Note that even though only the label prior,  $P(x)$  was assumed to be a MRF, the assumption of conditional independence of data implies that the posterior given in (4.1) is also a MRF. This allows one to reap the benefits of readily available tools of inference over a MRF. If the conditional independence assumption is not used, the posterior will usually not be a MRF making the inference difficult.

Now, if we turn our attention again toward the original aim, we are interested in classification of image sites. For classification purposes, we want to estimate the posterior over labels given the observations, i.e.,  $P(x|y)$ . In a generative framework, one expends efforts to model the joint distribution  $p(x,y)$ , which involves implicit modeling of the observations. In a discriminative framework, one models the distribution  $P(x|y)$  directly. A major advantage of doing this is that the true underlying generative model may be quite complex even though the class posterior is simple. This means that the generative approach may spend a lot of resources on modeling the generative models which are not particularly relevant to the task of inferring the class labels. Moreover, learning the class density models may become even harder when the training data is limited. The discriminative approach saves one from making simplistic assumptions about the data. This view forms the core theme of the model discussed in the following Sections.

### 4.3 Conditional Random Field (CRF)

Conditional Random Fields (CRFs) were originally proposed by Lafferty et al [15] in the context of segmentation and labeling of 1-D text sequences. CRFs are discriminative models that directly model the conditional distribution over labels

---

<sup>1</sup> With a slight abuse of notation, we will use the term 'MRF model' to indicate this posterior.

i.e.,  $P(x|y)$  as a Markov Random Field. This approach allows one to capture arbitrary dependencies between the observations without resorting to any model approximations. In this chapter, we will follow the generalized version of CRFs proposed by Kumar and Hebert [11] and [12]. They first introduced the extension of original 1-D CRFs to 2-D graphs over images. Their version also allows the use of arbitrary discriminative classifiers to model different types of interactions in labels and data, leading to more flexible and powerful generalization of CRFs.

We first restate the definition of CRFs as given by Lafferty et al. [13]. Let the observed data from an input image be given by  $y = \{y_i\}_{i \in S}$  where  $y_i$  is the data from  $i^{\text{th}}$  site and  $y_i \in \Re^c$ . The corresponding labels at the image sites are given by  $x = \{x_i\}_{i \in S}$ . First let us focus on binary classification problem, i.e.  $x_i \in \{-1, 1\}$ . Section 4.5.1 will describe its extension to multiclass labeling problem. The random variables  $x$  and  $y$  are jointly distributed, but in a discriminative framework, a conditional model  $P(x|y)$  is constructed from the observations and labels, and the marginal  $p(y)$  is not modeled explicitly.

**Definition 4.1. CRF:** Let  $G = (S, E)$  be a graph such that  $x$  is indexed by the vertices of  $G$ . Then  $(x, y)$  is said to be a conditional random field if, when conditioned on  $y$ , the random variables  $x_i$  obey the Markov property with respect to the graph:  $P(x_i|y, x_{S - \{i\}}) = P(x_i|y, x_{\mathcal{N}_i})$ , where  $S - \{i\}$  is the set of all the nodes in the graph except the node  $i$ ,  $\mathcal{N}_i$  is the set of neighbors of the node  $i$  in  $G$ , and  $x_\Omega$  represents the set of labels at the nodes in set  $\Omega$ .

Thus, a CRF is a random field globally conditioned on the observations  $y$ . The condition of positivity requiring  $P(x|y) > 0$ ,  $\forall x$  has been assumed implicitly. Using the Hammersley-Clifford theorem [6] and assuming only up to pairwise clique potentials to be nonzero, the conditional distribution over all the labels  $x$  given the observations  $y$  in a CRF can be written as,

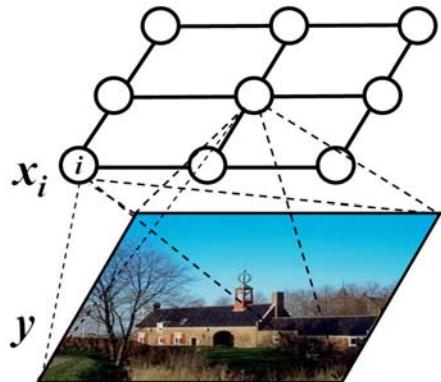
$$P(x|y) = \frac{1}{Z} \exp \left( \sum_{i \in S} A_i(x_i, y) + \sum_{i \in S} \sum_{j \in \mathcal{N}_i} I_{ij}(x_i, x_j, y) \right), \quad (4.2)$$

where  $Z$  is a normalizing constant known as the partition function, and  $-A_i$  and  $-I_{ij}$  are the unary and pairwise potentials respectively. With a slight abuse of notation, we will call  $A_i$  the *association potential* and  $I_{ij}$  the *interaction potential*.

There are two main differences between the conditional model given in Equation (4.2) and the traditional MRF framework given in Equation (4.1). First, in the conditional fields, the association potential at any site is a function of all the observations  $y$  while in MRFs (with the assumption of conditional independence of the data), the association potential is a function of data only at that site, i.e.,  $y_i$ . Second, the interaction potential for each pair of nodes in MRFs is a function of only labels, while in the conditional models it is a function of labels as well as all the observations  $y$ . As will be shown later, these differences play a crucial role in modeling arbitrary interactions in both observed data and labels in natural images in a principled manner.

In this discussion, we assume the random field given in Equation (4.2) to be homogeneous, i.e., the functional forms of  $A_i$  and  $I_{ij}$  are independent of the location  $i$ .

**Fig. 4.3** An illustration of a typical CRF for an example task of man-made structure detection in natural images. The aim is to label each site i.e., each  $16 \times 16$  image block whether it is a man-made structure or not. The top layer represents the labels on all the image sites. Note that each site  $i$  can potentially use features from the whole image  $y$  unlike the traditional MRFs.



In addition, we also assume the field to be isotropic implying that the label interactions are non-directional. In other words,  $I_{ij}$  is independent of the relative locations of sites  $i$  and  $j$ . Thus, subsequently we will drop the subscripts and simply use the notation  $A$  and  $I$  to denote the two potentials. In fact, the assumption of isotropy can be easily relaxed at the cost of a few additional parameters. Thus, we will consider models of the following form:

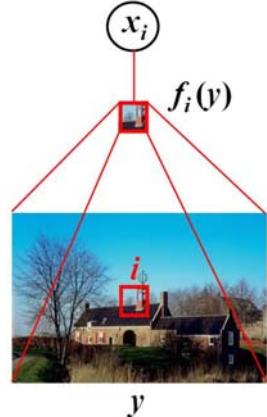
$$P(x|y) = \frac{1}{Z} \exp \left( \sum_{i \in S} A(x_i, y) + \sum_{i \in S} \sum_{j \in N_i} I(x_i, x_j, y) \right). \quad (4.3)$$

Due to this form of CRFs, it is possible to treat different applications from low-level image denoising to high-level contextual object detection seamlessly in a single framework. Figure 4.3 illustrates a typical CRF for an example image analysis task of man-made structure detection. Suppose, we are given an input image  $y$  shown in the bottom layer and we are interested in labeling each image site (in this case a  $16 \times 16$  image block) whether it contains a man-made structure or not. The top layer represents the labels  $x$  on all the image sites. Note that each site  $i$  can potentially use features from the whole image  $y$  unlike the traditional MRFs. In addition, CRFs allow to use image data to model interactions between two neighboring sites  $i$  and  $j$ . The following sections describe how the unary and the pairwise potentials are designed in CRFs.

### 4.3.1 Association Potential

In the CRF framework, the association potential,  $A(x_i, y)$ , can be seen as a measure of how likely a site  $i$  will take label  $x_i$  given image  $y$ , ignoring the effects of other sites in the image (Figure 4.4). Suppose,  $f(\cdot)$  is a function that maps an arbitrary patch in an image to a feature vector such that  $f : \mathcal{Y}_p \rightarrow \mathbb{R}^l$ . Here  $\mathcal{Y}_p$  is the set of all possible patches in all possible images. Let  $\omega_i(y)$  be an arbitrary patch in the neighborhood of site  $i$  in image  $y$  from which we want to extract a feature vector  $f(\omega_i(y))$ . Note that the neighborhood used for the patch  $\omega_i(y)$  need not be the same

**Fig. 4.4** Given a feature vector  $f_i(y)$  at site  $i$ , the association potential in CRFs can be seen as a measure of how likely the site  $i$  will take label  $x_i$ , ignoring the effects of other sites in the image. Note that the feature vector  $f_i(y)$  can be constructed by pooling arbitrarily complex dependencies in the observed data  $y$ .



as the label neighborhood  $\mathcal{N}_i$ . Indeed,  $\omega_i(y)$  can potentially be the whole image itself. For clarity, let us denote the feature vector  $f(\omega_i(y))$  at each site  $i$  by  $f_i(y)$ . The subscript  $i$  indicates the difference just in the feature vectors at different sites, *not* in the functional form of  $f(\cdot)$ . Then,  $A(x_i, y)$  is modeled using a local discriminative model that outputs the association of the site  $i$  with class  $x_i$  as,

$$A(x_i, y) = \log P'(x_i | f_i(y)), \quad (4.4)$$

where  $P'(x_i | f_i(y))$  is the local class conditional at site  $i$ . This form allows one to use an arbitrary domain-specific probabilistic discriminative classifier for a given task. This can be seen as a parallel to the traditional MRF models where one can use arbitrary local generative classifier to model the unary potential. One possible choice of  $P'(\cdot)$  is Generalized Linear Models (GLM), which are used extensively in statistics to model the class posteriors [18]. Logistic function is a commonly used link in GLMs although other choices such as probit link exist. Using a logistic function, the local class conditional can be written as,

$$P'(x_i=1 | f_i(y)) = \frac{1}{1+e^{-(w_0+w_1^T f_i(y))}} = \sigma(w_0 + w_1^T f_i(y)), \quad (4.5)$$

where  $w = \{w_0, w_1\}$  are the model parameters. This form of  $P'(\cdot)$  will yield a linear decision boundary in the feature space spanned by vectors  $f_i(y)$ . To extend the logistic model to induce a nonlinear decision boundary, a transformed feature vector at each site  $i$  can be defined as  $h_i(y) = [1, \phi_1(f_i(y)), \dots, \phi_R(f_i(y))]^T$  where  $\phi_k(\cdot)$  are arbitrary nonlinear functions. These functions can be seen as explicit kernel mapping of the original feature vector into a high dimensional space. The first element of the transformed vector is kept as 1 to accommodate the bias parameter  $w_0$ . Further, since  $x_i \in \{-1, 1\}$ , the probability in Equation (4.5) can be compactly expressed as,

$$P'(x_i | y) = \sigma(x_i w^T h_i(y)). \quad (4.6)$$

Finally, for this choice of  $P'(\cdot)$ , the association potential can be written as,

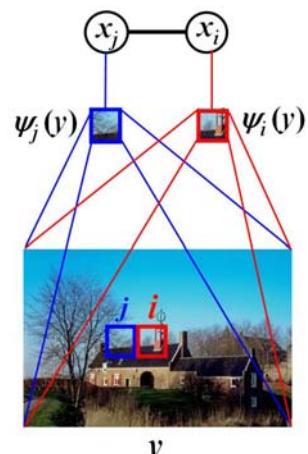
$$A(x_i, y) = \log(\sigma(x_i w^T h_i(y))) \quad (4.7)$$

This transformation ensures that the CRF is equivalent to a logistic classifier if the interaction potential in Equation (4.3) is set to zero. Besides GLMs, discriminative classifiers based on SVM, Neural Network and Boosting have been successfully used in modeling association potential in the literature. Note that in Equation (4.7), the transformed feature vector at each site  $i$  i.e.,  $h_i(y)$  is a function of the whole set of observations  $y$ . This allows one to pool arbitrarily complex dependencies in the observed data for the purpose of classification. On the contrary, the assumption of conditional independence of the data in the traditional MRF framework allows one to use data only from a particular site, i.e.,  $y_i$ , to design the log-density, which acts as the association potential as shown in Equation (4.1).

### 4.3.2 Interaction Potential

In the CRF framework, the interaction potential can be seen as a measure of how the labels at neighboring sites  $i$  and  $j$  interact given the observed image  $y$  (Figure 4.5). To model the interaction potential,  $I(\cdot)$ , we first analyze a form commonly used in the MRF framework. For the isotropic, homogeneous Ising model, the interaction potential is given as  $I(\cdot) = \beta x_i x_j$ , which penalizes every dissimilar pair of labels by the cost  $\beta$  [8]. This form of interaction favors piecewise constant smoothing of the labels without considering the discontinuities in the observed data explicitly. Geman and Geman extended the Ising model to a line-process model which allows discontinuities in labels through piecewise continuous smoothing [4].

**Fig. 4.5** Given feature vectors  $\psi_i(y)$  and  $\psi_j(y)$  at two neighboring sites  $i$  and  $j$  respectively, the interaction potential can be seen as a measure of how the labels at sites  $i$  and  $j$  influence each other. Note that such interaction in labels is dependent on the observed image data  $y$ , unlike the traditional generative MRFs.



However, such discontinuity adaptive models also do not use the observed data to model the discontinuities.

In contrast, in the CRF formulation, the interaction potential is a function of all the observations  $y$ . Suppose,  $\psi(\cdot)$  is a function that maps an arbitrary patch in an image to a feature vector such that  $\psi : \mathcal{Y}_p \rightarrow \mathbb{R}^q$ . Let  $\Omega_i(y)$  be an arbitrary patch in the neighborhood of site  $i$  in image  $y$  from which we want to extract a feature vector  $\psi(\Omega_i(y))$ . Note that the neighborhood used for the patch  $\Omega_i(y)$  need not be the same as the label neighborhood  $\mathcal{N}_i$ . For clarity, let us denote the feature vector  $\psi(\Omega_i(y))$  at each site  $i$  by  $\psi_i(y)$ . Similarly, we define a feature vector  $\psi_j(y)$  for site  $j$ . Again, to emphasize, the subscripts  $i$  and  $j$  indicate the difference just in the feature vectors at different sites, *not* in the functional form of  $\psi(\cdot)$ . Given the features at two different sites, we want to learn a pairwise discriminative model  $P''(x_i=x_j|\psi_i(y), \psi_j(y))$ . Note that by choosing the function  $\psi_i$  to be different from  $f_i$ , used in Equation (4.5), information different from  $f_i$  can be used to model the relations between pairs of sites.

For a pair of sites  $(i, j)$ , let  $\mu_{ij}(\psi_i(y), \psi_j(y))$  be a new feature vector such that  $\mu_{ij} : \mathbb{R}^q \times \mathbb{R}^q \rightarrow \mathbb{R}^q$ . Denoting this feature vector as  $\mu_{ij}(y)$  for simplification, the interaction potential is modeled as,

$$I(x_i, x_j, y) = x_i x_j v^T \mu_{ij}(y), \quad (4.8)$$

where  $v$  are the model parameters. Note that the first component of  $\mu_{ij}(y)$  is fixed to be 1 to accommodate the bias parameter. There are two interesting properties of the interaction potential given in Equation (4.8). First, if the association potential at each site and the interaction potentials for all the pairwise cliques except the pair  $(i, j)$  are set to zero in Equation (4.3), the CRF acts as a logistic classifier which yields the probability of the site pair to have the same labels given the observed data. Of course, one can generalize the form in Equation (4.8) as,

$$I(x_i, x_j, y) = \log P''(x_i, x_j | \psi_i(y), \psi_j(y)), \quad (4.9)$$

similar to the association potential defined in Section 4.3.1 and can use arbitrary pairwise discriminative classifier to define this term. The second property of the interaction potential form given in Equation (4.8) is that it generalizes the Ising model. The original Ising form is recovered if all the components of vector  $v$  other than the bias parameter are set to zero in Equation (4.8). A geometric interpretation of interaction potential is that it partitions the space induced by the relational features  $\mu_{ij}(y)$  between the pairs that have the same labels and the ones that have different labels. Hence Equation (4.8) acts as a data-dependent discontinuity adaptive model that will moderate smoothing when the data from the two sites is 'different'. The data-dependent smoothing can especially be useful to absorb the errors in modeling the association potential. Anisotropy can be easily included in the CRF model by parameterizing the interaction potentials of different directional pairwise cliques with different sets of parameters  $v$ .

## 4.4 Parameter Learning and Inference

One of the crucial requirements to make the CRF-based models applicable to a variety of real-world tasks is accurate and efficient parameter learning in these models. Here, we focus on maximum likelihood based supervised learning of CRFs.

For 1-D sequential CRFs proposed by Lafferty et al. [15], exact maximum likelihood parameter learning is feasible because the induced graph does not contain loops. However, when a graph contains loops, it is generally infeasible to exactly maximize the likelihood with respect to the parameters. Therefore, a critical issue in applying CRFs to image-based applications is the design of effective parameter learning techniques that can operate on arbitrary graphs.

### 4.4.1 Maximum Likelihood Parameter Learning

Let  $\theta$  be the set of unknown CRF parameters where  $\theta = \{w, v\}$ . Given  $M$  i.i.d. labeled training images, the maximum likelihood estimates of the parameters are given by maximizing the log-likelihood  $l(\theta) = \sum_{m=1}^M \log P(x^m | y^m, \theta)$ , i.e.,

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{m=1}^M \left\{ \sum_{i \in S^m} \log \sigma(x_i^m w^T h_i(y^m)) + \sum_{i \in S^m} \sum_{j \in \mathcal{N}_i} x_i^m x_j^m v^T \mu_{ij}(y^m) - \log Z^m \right\}, \quad (4.10)$$

where the partition function for the  $m^{th}$  image is,

$$Z^m = \sum_x \exp \left\{ \sum_{i \in S^m} \log \sigma(x_i w^T h_i(y^m)) + \sum_{i \in S^m} \sum_{j \in \mathcal{N}_i} x_i x_j v^T \mu_{ij}(y^m) \right\}.$$

Note that  $Z^m$  is a function of the parameters  $\theta$  and the observed data  $y^m$ . For learning the parameters using gradient ascent, the derivatives of the log-likelihood are

$$\frac{\partial l(\theta)}{\partial w} = \frac{1}{2} \sum_m \sum_{i \in S^m} (x_i^m - \langle x_i \rangle_{\theta; y^m}) h_i(y^m), \quad (4.11)$$

$$\frac{\partial l(\theta)}{\partial v} = \sum_m \sum_{i \in S^m} \sum_{j \in \mathcal{N}_i} (x_i^m x_j^m - \langle x_i x_j \rangle_{\theta; y^m}) \mu_{ij}(y^m). \quad (4.12)$$

Here  $\langle \cdot \rangle_{\theta; y^m}$  denotes expectation with  $P(x|y^m, \theta)$ . Ignoring  $\mu_{ij}(y^m)$ , gradient ascent with Equation (4.12) resembles the problem of learning in Boltzmann machines.

For arbitrary graphs with loops, the expectations in Equation (4.11) and Equation (4.12) cannot be computed exactly due to the combinatorial size of the label space. Sampling procedures such as Markov Chain Monte Carlo (MCMC) can be used to approximate the true expectations. Unfortunately, MCMC techniques have two main problems: a long ‘burn-in’ period (which makes them slow) and high variance in estimates. Although several techniques have been suggested to approximate the

expectations, let us focus on two popular methods (see [10] for other choices and a detailed comparison).

#### 4.4.1.1 Pseudo-Marginal Approximation (PMA)

It is easy to see that if we had true marginal distributions  $P_i(x_i|y, \theta)$  at each site  $i$ , and  $P_{ij}(x_i, x_j|y, \theta)$  at each pair of sites  $i$  and  $j \in \mathcal{N}_i$ , we could compute exact expectations as:

$$\langle x_i \rangle_{\theta; y} = \sum_{x_i} x_i P_i(x_i|y, \theta) \quad \text{and} \quad \langle x_i x_j \rangle_{\theta; y} = \sum_{x_i, x_j} x_i x_j P_{ij}(x_i, x_j|y, \theta).$$

Since computing exact marginal distributions is in general infeasible, a standard approach is to replace the actual marginals by pseudo-marginals. For instance, one can use loopy Belief Propagation (BP) to get these pseudo-marginals. It has been shown in practice that for many applications loopy BP provides good estimates of the marginals.

#### 4.4.1.2 Saddle Point Approximation (SPA)

In Saddle Point Approximation (SPA), one makes a discrete approximation of the expectations by directly using best estimates of labels at a given setting of parameters. This is equivalent to approximating the partition function ( $Z$ ) such that the summation over all the label configurations  $x$  in  $Z$  is replaced by the largest term in the sum, which occurs at the most probable label configuration. In other words, if

$$\hat{x} = \arg \max_x P(x|y, \theta),$$

then according to SPA,

$$Z \approx \exp \left\{ \sum_{i \in S} \log \sigma(\hat{x}_i w^T h_i(y)) + \sum_{i \in S} \sum_{j \in \mathcal{N}_i} \hat{x}_i \hat{x}_j v^T \mu_{ij}(y) \right\}.$$

This leads to a very simple approximation to the expectation, i.e.,  $\langle x_i \rangle_{\theta; y} \approx \hat{x}_i$ . If we further assume mean-field type decoupling, i.e.,  $\langle x_i x_j \rangle_{\theta; y} = \langle x_i \rangle_{\theta; y} \langle x_j \rangle_{\theta; y}$ , it also follows that  $\langle x_i x_j \rangle_{\theta; y} \approx \hat{x}_i \hat{x}_j$ . Readers familiar with perceptron learning rules can readily see that with such an approximation, the updates in Equation (4.11) are very similar to perceptron updates.

However, this discrete approximation raises a critical question: Will the gradient ascent of the likelihood with such gradients converge? It has been shown empirically that while the approximate gradient ascent is not strictly convergent in general, it is weakly convergent in that it oscillates within a set of good parameters, or converges to a good parameter with isolated large deviations. In fact one can show that this weak-convergence behavior is tied to the empirical error of the model [10]. To pick a good parameter setting, one can use any of the popular heuristics used for perceptron learning with inseparable data. For instance, one can let the algorithm run up to

some fixed number of iterations and pick the parameter setting that minimizes the empirical error. Even though lack of strict convergence can be seen as a drawback of SPA, the main advantage of these methods is very fast learning of parameters with performance similar to or better than pseudo-marginal methods.

#### 4.4.2 Inference

Given a new test image  $y$ , the problem of inference is to find the optimal labels  $x$  over the image sites, where optimality is defined with respect to a given cost function. Maximum A Posteriori (MAP) solution is a widely used estimate that is optimal with respect to the zero-one cost function defined as,

$$C(x, x^*) = 1 - \delta(x - x^*), \quad (4.13)$$

where  $x^*$  is the true label configuration, and  $\delta(x - x^*)$  is 1 if  $x = x^*$ , and 0 otherwise. The MAP solution is defined as,

$$\hat{x} = \arg \max_x P(x|y, \theta).$$

For binary classifications, the MAP estimate can be computed exactly for an undirected graph using the max-flow/min-cut type of algorithms if the probability distribution meets certain conditions [5]. While using the Ising MRF model, exact MAP solution can be computed if  $\beta_m \geq 0$ . For the CRF model, since max-flow algorithms do not allow negative interaction between the sites, the data-dependent smoothing for each clique is set to be  $v^T \mu_{ij}(y) = \max\{0, v^T \mu_{ij}(y)\}$ , yielding an approximate MAP solution.

An alternative to the MAP solution is the Maximum Posterior Marginal (MPM) solution which is optimal for the sitewise zero-one cost function defined as,

$$C(x, x^*) = \sum_{i \in S} (1 - \delta(x_i - x_i^*)), \quad (4.14)$$

where  $x_i^*$  is the true label at the  $i^{th}$  site. The MPM solution at each site is defined as,

$$\hat{x}_i = \arg \max_{x_i} P_i(x_i|y, \theta), \quad \text{where } P_i(x_i|y, \theta) = \sum_{x - x_i} P(x|y, \theta),$$

and  $x - x_i$  denotes all the node variables except for node  $i$ . The MPM computation requires marginalization over a large number of variables which is generally NP-hard. However, as discussed before, one can use loopy BP to obtain an estimate of the MPM solution.

#### 4.5 Extensions

A large number of extensions of the basic binary CRFs have been proposed in the literature. In the following sections, we discuss two key extensions: Multiclass CRFs

to deal with multiclass labeling problems, and Hierarchical CRFs to incorporate hierarchical context in the model.

#### 4.5.1 Multiclass CRF

There are several applications in computer vision that require the nodes in the graph to take multiple class labels. For example, in semantic scene segmentation task, one may want to assign each pixel into one of many classes such as *sky*, *water*, *grass*, etc. In the case of image denoising applied to a 256 gray-level image, each pixel may take up to 256 labels. In the part-based paradigm of object detection, usually there are more than two characteristic parts that make the full object, and the goal is to label each generic part in the scene as a specific part of the object or background.

The extension of binary CRFs to the multiclass case is relatively straightforward. The only difference in multiclass CRF formulation is that the labels at the image sites are given by  $x = \{x_i\}_{i \in S}$ , where  $x_i \in \{1, \dots, C\}$  and  $C$  is the number of classes. To illustrate various terms in the model, we will take the example of parts-based object detection, in which, each image site is a part and the first  $(C - 1)$  labels correspond to specific object parts and the  $C^{th}$  label corresponds to the background class.

Following the arguments given in Section 4.3.1 and the form of the association potential for binary CRFs (Equation 4.7), the association potential can be easily generalized to the multiclass case as,

$$A(x_i, y) = \sum_{k=1}^C \delta(x_i = k) \log P'(x_i = k|y), \quad (4.15)$$

where  $\delta(x_i = k)$  is 1 if  $x_i = k$  and 0 otherwise. Let  $h_i(y)$  be a (possibly kernelized) feature vector at each site  $i$ . Note that, in the case of object detection, the vector  $h_i(y)$  encodes the appearance based features of the  $i^{th}$  part. To model  $P'(x_i = k|y)$ , one can simply use the multiclass version of the logistic function described for the binary CRFs in Section 4.3.1. This leads to the softmax function in the multiclass case where,

$$P'(x_i = k|y) = \begin{cases} \frac{\exp(w_k^T h_i(y))}{1 + \sum_{l=1}^{C-1} \exp(w_l^T h_i(y))} & \text{if } k < C \\ \frac{1}{1 + \sum_{l=1}^{C-1} \exp(w_l^T h_i(y))} & \text{if } k = C. \end{cases} \quad (4.16)$$

Here,  $w_k$  are the model parameters for  $k = 1 \dots C - 1$ . For a  $C$  class classification problem, one needs only  $C - 1$  independent hyperplanes, which may lie in a high dimensional (kernel-projected) space inducing a non-linear decision boundary in the original feature space. In the application of object detection, the association potential discriminatively models the individual appearance of each part in the image.

The interaction potential in CRFs predicts how the labels at two sites interact given the observations. Generalizing the interaction potential given for binary CRFs, interaction potential for multiclass CRFs can be written as,

$$I(x_i, x_j, y) = \sum_{k=1}^C \sum_{l=1}^C v_{kl}^T \mu_{ij}(y) \delta(x_i = k) \delta(x_j = l). \quad (4.17)$$

Here,  $\mu_{ij}(y)$  is the pairwise relational vector for a site pair  $(i, j)$ , and  $v_{kl}$  are the model parameters. Note that in the case of object detection, the vector  $\mu_{ij}(y)$  encodes the pairwise features required for forcing geometric and possibly photometric consistency in the pair of parts. For undirected graphs, the site pairs are unordered sets implying that  $v_{kl} = v_{lk}$  for  $k, l = 1 \dots C$ . The from of interaction potential given in Equation (4.17) is a generalization of the Potts model used commonly in computer vision problems such as image segmentation and restoration. The standard Potts model can be recovered from Equation (4.17) if  $v_{kl} = 0$  when  $k \neq l$ , and all the elements of the vector  $v_{kl}$  are set to zero except the bias term. A more specific but popular form of Potts model is achieved if the bias terms for all the vectors  $v_{kk} \forall k$  are also fixed to be the same. Similar to the interaction potential of the binary CRF, multiclass interaction potential can be seen as a pairwise discriminative model which partitions the pairwise relational feature space (induced by the features  $\mu_{ij}(y)$ ) in  $C(C + 1)/2$  regions.

It is important to note that, to enforce the geometric consistency relationship between parts, the interaction between part labels has to use observed data (e.g. the location of patches). Since, the pairwise potential  $I$  is a function of observed data in CRFs, these fields provide a principled way to represent relations between parts in a random-field framework.

Let  $\theta$  be the set of CRF parameters where  $\theta = \{w_k\}_{k=1,\dots,C-1}, \{v_{kl}\}_{k,l=1,\dots,C}\}$ . To learn  $\theta$  via maximum likelihood, similar to the binary CRFs, one can write the gradient of log-likelihood as,

$$\frac{\partial l(\theta)}{\partial w_k} = \sum_m \sum_{i \in S^m} \left( \delta(x_i^m = k) - \langle \delta(x_i = k) \rangle \right) h_i(y^m), \quad (4.18)$$

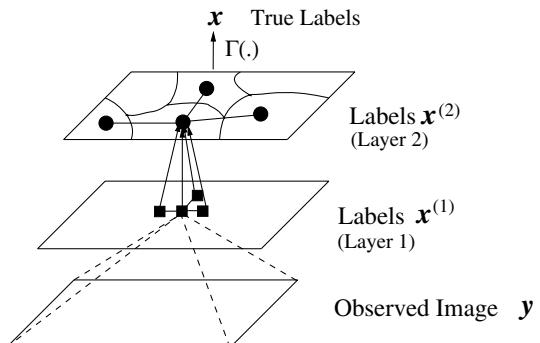
$$\frac{\partial l(\theta)}{\partial v_{kl}} = \sum_m \sum_{i \in S^m} \sum_{j \in \mathcal{N}_i} \left( \delta(x_i^m = k) \delta(x_j^m = l) - \langle \delta(x_i = k) \delta(x_j = l) \rangle \right) \mu_{ij}(y^m), \quad (4.19)$$

where  $\langle . \rangle$  denotes expectation with respect to the distribution  $P(x|y^m, \theta)$  and  $m$  indexes over the training images. Generally the expectations in Equation (4.18) and Equation (4.19) cannot be computed exactly even for moderate-size problems. Similar to the previous discussion in Section 4.4, these expectations can be approximated by either pseudo-marginals or Saddle Point Approximation with multiclass extensions of min-cuts [3]. Similarly, for inference, one can get the labels either using approximate MAP obtained by multiclass min-cut or using approximate MPM via loopy BP.

### 4.5.2 Hierarchical CRF

So far, we have discussed spatial interactions in natural images at pixel, block or patch level for binary or multiclass classification problems. However, in natural

**Fig. 4.6** A simple illustration of a two-layer hierarchical field for contextual classification. Squares and circles represent sites at the two layers. Only one node along with its neighbors is shown for each layer for clarity. Layer 1 models short-range interactions while layer 2 models long range dependencies in images. The true labels  $x$  are obtained from the top layer by a simple replication mapping  $\Gamma(\cdot)$ .



images, there are different levels of context one would like to use to improve classification accuracy. For instance, for pixelwise image labeling problem, the local smoothness of pixel labels provides local context. On the other hand, there exists a higher level global context since image regions follow probable configurations. For example, sky tends to occur above water or vegetation. Similarly, for the problem of parts-based object detection, local context is the geometric relationship among parts of an object while the relative spatial configurations of different objects (e.g., monitor, keyboard and mouse) provides the global context. Here we present a high-level discussion on how one can use hierarchy of CRFs to improve classification in images. For a detailed discussion on this topic, see [13].

A simple two-layer hierarchical model is shown in Figure 4.6, in which each layer is modeled as a separate CRF. The first layer models short range interactions among the sites such as label smoothing for pixelwise labeling, or geometric consistency among parts of an object. The second layer models the long range interactions between groups of sites corresponding to different coherent regions or objects. Thus, this layer can take into account interactions between different objects (monitor/keyboard) or regions (sky/water).

The two layers of the hierarchy are coupled with directed links. A node in layer 1 may represent a single pixel or a patch while a node in layer 2 represents a larger homogeneous region or a whole object. Each node in the two layers is connected to its neighbors through undirected links. In addition, each node in layer 2 is also connected to multiple nodes in layer 1 through directed links. The use of directed links between the two layers, instead of the undirected ones, avoids the intractability of dealing with a large partition function. Each layer being a CRF, any node in layer 1 can potentially use arbitrary features from the whole image. The top layer uses the output of layer 1 as input through the directed links.

Given the observed data  $y = \{y_i\}_{i \in S}$  in an image, we are interested in finding the labels,  $x = \{x_i\}_{i \in S}$ , where  $x_i \in \mathcal{L}$  and  $|\mathcal{L}|$  is the number of classes. For image labeling, a site is a pixel and a class may be *sky*, *grass*, etc., while for contextual

object detection, a site is a patch and a class may refer to objects (e.g., *keyboard* or *mouse*). The set of sites in layer 1 is  $S^{(1)}$  such that  $S^{(1)} = S$ , while that in layer 2 is denoted by  $S^{(2)}$ . The nodes in layer 2 induce a partition over the set  $S^{(1)}$  such that a subset of nodes in layer 1 correspond to one node in layer 2. Formally, a partition  $h$  is defined as  $h : S^{(1)} \rightarrow S^{(2)}$  such that, if  $S_r^{(1)}$  is a subset of nodes in layer 1 corresponding to node  $r \in S^{(2)}$ , then  $S^{(1)} = \bigcup_r S_r^{(1)}$  and  $S_r^{(1)} \cap S_s^{(1)} = \emptyset \quad \forall r, s \in S^{(2)}$ . Let the space of all partitions be denoted as  $\mathcal{H}$ . This partition should not be confused with an image partition, since it is defined over the sites in  $S^{(1)}$ , which may not correspond to the image pixels (e.g., in object detection, where sites are random image patches). Let the labels on the sites in the two layers be given by  $x^{(1)} = \{x_i^{(1)}\}_{i \in S^{(1)}}$  and  $x^{(2)} = \{x_r^{(2)}\}_{r \in S^{(2)}}$ , where  $x_i^{(1)} \in \mathcal{L}^{(1)}$  and  $x_r^{(2)} \in \mathcal{L}^{(2)}$ , where  $\mathcal{L}^{(2)} = \mathcal{L}$ . The nodes in layer 1 may take pseudo-labels that are different from the final desired labels. For instance, in object detection, a node at layer 1 may be labeled as 'a certain part' of an object rather than the object itself. In fact, the labels at this layer can be seen as noisy versions of the true desired labels.

Given an image  $y$ , we are interested in obtaining the discriminative distribution  $P(x|y)$  over the true labels. Let us define a space of valid partitions,  $\mathcal{H}_v$ , such that  $\forall h \in \mathcal{H}_v, x_i = x_r^{(2)} \quad \forall i \in S_r^{(1)}$ , where  $r = h(i)$ . This implies that multiple nodes in layer 1 form a hypothesis about a single *homogeneous* region or an object in layer 2. Further, we define a replication mapping,  $\Gamma(\cdot)$ , which takes any value (discrete or continuous) on node  $r$  and assigns it to all the nodes in  $S_r^{(1)}$ . Thus, given a partition  $h \in \mathcal{H}_v$ , and the corresponding labels  $x^{(2)}$ , the labels  $x$  can be obtained simply by replication. This implies,  $P(x|y) \equiv P(x^{(2)}|h, y)$  if  $h \in \mathcal{H}_v$ . However, given an observed image  $y$ , the constraint  $h \in \mathcal{H}_v$  is too restrictive. Instead, one can define a distribution,  $P(h|y)$ , that prefers partitions in  $\mathcal{H}_v$  over all possible partitions, and,

$$\begin{aligned} P(x|y) &\cong \sum_{h \in \mathcal{H}} P(x^{(2)}|h, y)P(h|y) \\ &= \sum_{h \in \mathcal{H}} \sum_{x^{(1)}} P(x^{(2)}|h, x^{(1)})P(h|x^{(1)})P(x^{(1)}|y), \end{aligned} \quad (4.20)$$

where both  $P(x^{(1)}|y)$  and  $P(x^{(2)}|h, x^{(1)})$  are modeled as CRFs. In Equation (4.20), computing the sum over all the possible configurations of  $x^{(1)}$  is a NP-hard problem. One naive way to reduce the complexity is to do inference in layer 1 until equilibrium is reached and then use this configuration  $\hat{x}^{(1)}$  as input to the next layer, i.e.,  $P(x^{(1)}|y) = \delta(x^{(1)} - \hat{x}^{(1)})$ . However, by doing this, one loses the power of modeling the uncertainty associated with the labels in layer 1, which was included explicitly in Equation (4.20) through  $P(x^{(1)}|y)$ . Here, we discuss a simple variant, where along with the equilibrium configuration, one also propagates the uncertainty associated with it to the next layer. The sitewise maximum marginal configuration are used as  $\hat{x}^{(1)}$ . Let the marginals at each site  $i$  be  $b_i(x_i^{(1)}) = \sum_{x^{(1)}} P(x^{(1)}|y)$ , and  $b(x^{(1)}) = \{b_i(x_i^{(1)})\}_{i \in S^{(1)}}$ . The belief set,  $b(x^{(1)})$  is propagated as an input to the next

layer. Since the configuration  $\hat{x}^{(1)}$  can be obtained directly from  $b(x^{(1)})$  by taking its sitewise maximum configuration, we omit explicit conditioning on  $\hat{x}^{(1)}$ . Thus,

$$P(x|y) \approx \sum_{h \in \mathcal{H}} P(x^{(2)}|h, b(x^{(1)})) P(h|b(x^{(1)})). \quad (4.21)$$

Note that both terms in the summation implicitly include the transition probabilities  $P(x_r^{(2)}|\hat{x}_i^{(1)})$ . For the first term, these are absorbed in the unary potential of the CRF in layer 2.

The model describing layer 1 is a simple multiclass CRF and different potentials are designed as discussed before in Section 4.5.1. The CRF formulation for layer 2 can be obtained in the same way except that the observations for this layer are  $b(x^{(1)})$ , the set of sites is  $S^{(2)}$ , and the label set is  $\mathcal{L}^{(2)}$ . The only difference lies in the form of the unary potential. Each node  $r \in S^{(2)}$  in this layer receives beliefs as input from the nodes contained in set  $S_r^{(1)}$  from the layer below. Taking into consideration the transition probabilities on the directed links between node  $r$  and all the nodes in  $S_r^{(1)}$ , the unary potential can be written as,

$$\begin{aligned} A^{(2)}(x_r^{(2)}, b(x^{(1)})) = & \sum_{k \in \mathcal{L}^{(2)}} \left\{ \delta(x_r^{(2)} = k) \right. \\ & \left. \left( \log P'(x_r^{(2)} = k | b(x^{(1)})) + \frac{1}{|S_r^{(1)}|} \sum_{i \in S_r^{(1)}} \log P(x_r^{(2)} = k | \hat{x}_i^{(1)}) \right) \right\}. \end{aligned} \quad (4.22)$$

Here, the first term in parentheses on the right hand side involves local classifier  $P'(\cdot)$ , which is again modeled as a softmax function. It takes features as input, which are constructed using the beliefs  $b(x^{(1)})$  at layer 1. The second term arises due to the directed connections between each node  $r \in S^{(2)}$  in layer 2 to all the nodes in the set  $S_r^{(1)}$  in layer 1. The effect of this term can be understood by switching the first term off along with the interaction potential. This will lead to the intuitive reasoning of assigning node  $r$  that label which maximizes the joint transition probability (computed by assuming each site in  $S_r^{(1)}$  to be independent) given a label configuration  $\hat{x}^{(1)}$  at layer 1. The term,  $|S_r^{(1)}|$  acts as a normalizer that takes into account the different cardinalities of sets  $S_r^{(1)}$ . In the interaction potential for this layer, the features  $\mu_{ij}(\cdot)$  are designed such that they capture relative configurations of two regions or objects.

The distribution  $P(h|b(x^{(1)}))$  indicates *goodness* of a partition in layer 2. Here, we just mention that one can design this function according to the application domain. One can find more details on possible choices in [13]. The set of parameters  $\Theta$ , to be learned in the hierarchical model, includes the parameters of the CRFs at layer 1 and layer 2, and the transition probability matrices  $P(x_r^{(2)}|\hat{x}_i^{(1)})$ . The CRF parameters for each layer are the parameters of the unary and pairwise potentials i.e.,  $\theta^{(\alpha)} = \left\{ w_k^{(\alpha)}, v_{kl}^{(\alpha)} \right\}_{\forall k,l}^{\alpha=1,2}$ . The parameters in the joint model are learned sequentially using loopy BP. The procedure is a simple extension of learning in multiclass

CRFs discussed before. Similarly, inference in this model is carried out using a combination of loopy BP and sampling of partitions. We refer the reader to [13] for more details on learning and inference.

## 4.6 Applications

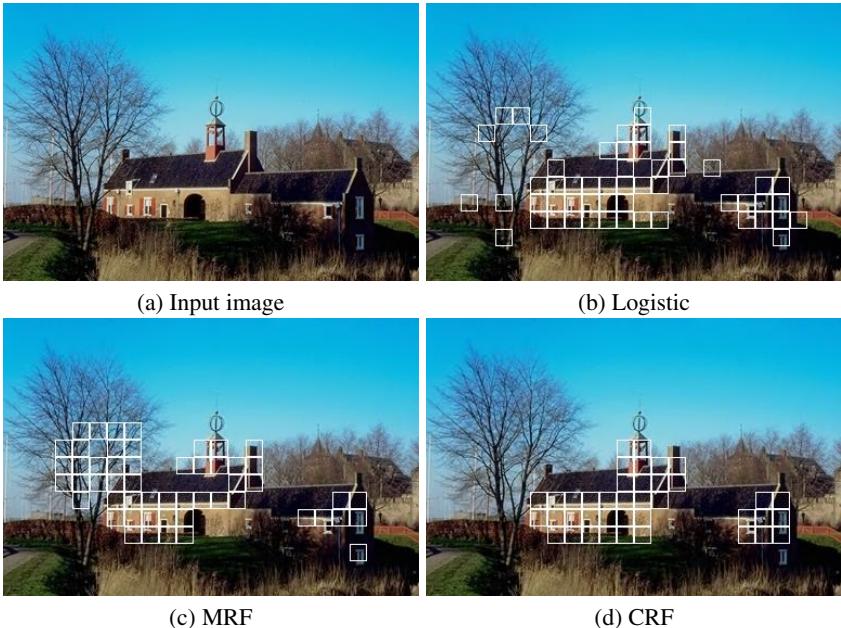
In this Section, we discuss a few real-world applications of different types of CRFs. The application of binary CRFs is considered on man-made structure detection, while the performance of multiclass and hierarchical CRFs is tested on image classification and contextual object detection tasks.

### 4.6.1 Man-Made Structure Detection

Detecting man-made structures in natural scenes is difficult because there are significant within class variations in the appearance of data from the *structured* class. Similarly, the data from the *nonstructured* class is virtually unconstrained, and there is a large overlap between these two classes. The training and the test set used in this study contained 108 and 129 images respectively from the Corel image database. Each image is divided into nonoverlapping  $16 \times 16$  pixels blocks. Each block forms a site in the graph. The whole training set contained 36,269 blocks from the *nonstructured* class, and 3,004 blocks from the *structured* class.

To generate the features, the intensity gradients contained within a window in the image are combined to yield a histogram over gradient orientations. Each histogram count is weighted by the gradient magnitude at that pixel and smoothed using kernel smoothing. Heaved central-shift moments are computed to capture the the average *spikiness* of the smoothed histogram as an indicator of the *structuredness* of the patch. The orientation based feature is obtained by passing the absolute difference between the locations of the two highest peaks of the histogram through sinusoidal nonlinearity. The absolute location of the highest peak is also used.

For each image, two different type of feature vectors at each site are computed. First a *single-site* feature vector at the site  $i$ ,  $s_i(y_i)$  is computed using the histogram from the data  $y_i$  at that site. Obviously, this vector does not take into account the influence of the data in the neighborhood of that site. The vector  $s_i(y_i)$  is composed of the first three moments and the two orientation based features described above. Next, a *multiscale* feature vector at the site  $i$ ,  $f_i(y)$  is computed which explicitly takes into account the dependencies in the data contained in the neighboring sites. It should be noted that the neighborhood for the data interaction need not be the same as for the label interaction. To compute  $f_i(y)$ , smoothed histograms are obtained at three different scales, where each scale is defined as a varying window size around the site  $i$ . The number of scales is chosen to be 3, with the scales changing in regular octaves. The lowest scale is fixed at  $16 \times 16$  pixels (i.e., the size of a single site), and the highest scale at  $64 \times 64$  pixels. The moment and orientation based features are obtained at each scale similar to  $s_i(y_i)$ . In addition, two interscale features are also obtained using the highest peaks from the histograms at consecutive scales. To avoid



**Fig. 4.7** Structure detection results on a test example for different methods. For similar detection rates, CRF reduces the false positives considerably.

redundancy in the moments based features, only two moment features are used from each scale yielding a 14 dimensional feature vector.

To make the unary classifier in the CRF more powerful, a transformed feature vector  $h_i(y)$  is computed at each site  $i$  by using an explicit quadratic kernel. The quadratic mapping gives a 119 dimensional vector at each site. The features  $\psi_i$  are chosen to be the same as  $f_i$ . Further, the pairwise data vector  $\mu_{ij}(y)$  is obtained by concatenating  $\psi_i(y)$  and  $\psi_j(y)$ . The parameters of the CRF model were learned using the maximum likelihood framework as described before.

#### 4.6.1.1 Qualitative Evaluation

For an input test image given in Figure 4.7 (a), the *structure* detection results from three methods are shown in Figure 4.7. The blocks identified as *structured* have been shown enclosed within an artificial boundary. It can be noted that for similar detection rates, the number of false positives have significantly reduced for the CRF based detection. Locally, different branches may yield features similar to those from the man-made structures. The logistic classifier does not enforce smoothness in labels, which led to increased isolated false positives. However, the MRF solution with Ising model simply smooths the labels without taking observations into account resulting in a smoothed false positive region around the tree branches.



**Fig. 4.8** Detection of a building in poor illumination conditions in a test image. CRFs can improve the detection rate while simultaneously reducing the false positive rate.

The performance of MRF and CRF is compared on another test example requiring detection of a building in poor illumination conditions (Figure 4.8). CRFs give higher detection rate while reducing the false positive rate by enforcing interactions among the labels as well as the data from multiple scales.

#### 4.6.1.2 Quantitative Evaluation

To carry out the quantitative evaluations, the detection rates and the number of false positives per image for each technique are compared. To avoid the confusion due to different effects in the CRF model, the first set of experiments is conducted using the *single-site* features for all the three methods. Thus, no neighborhood data interaction is used for both the logistic classifier and the CRF, i.e.,  $f_i(y) = s_i(y)$ . The comparative results for the three methods are given in Table 4.1 next to 'MRF', 'Logistic-' and 'CRF-'. For comparison purposes, the false positive rate of the logistic classifier is fixed to be the same as the CRF in all the experiments. It can be noted that for similar false positives, the detection rates of the traditional MRF and the CRF are higher than the logistic classifier due to the label interaction. However, the higher detection rate of the CRF in comparison to the MRF indicates the gain due to the use of discriminative models in the association and interaction potentials. In the next experiment, to take advantage of the power of the CRF framework, data interaction was allowed for both the logistic classifier as well as the CRF ('Logistic' and 'CRF' in Table 4.1). The CRF detection rate increases substantially and the false positives decrease further indicating the importance of allowing the data interaction in addition to the label interaction.

#### 4.6.2 Image Classification and Contextual Object Detection

The experiments in this Section demonstrate the capability of hierarchical CRFs. The first set of experiments is based on the 'Beach' dataset from [24], which contains a collection of consumer photographs. The goal was to assign to each image pixel one of 6 class labels: {sky, water, sand, skin, grass, other}. This dataset is

**Table 4.1** Detection Rates (DR) and False Positives (FP) for the test set containing 129 images (49,536 sites). FP for logistic classifier were kept to be the same as for CRF for DR comparison. Superscript '−' indicates no neighborhood data interaction was used.

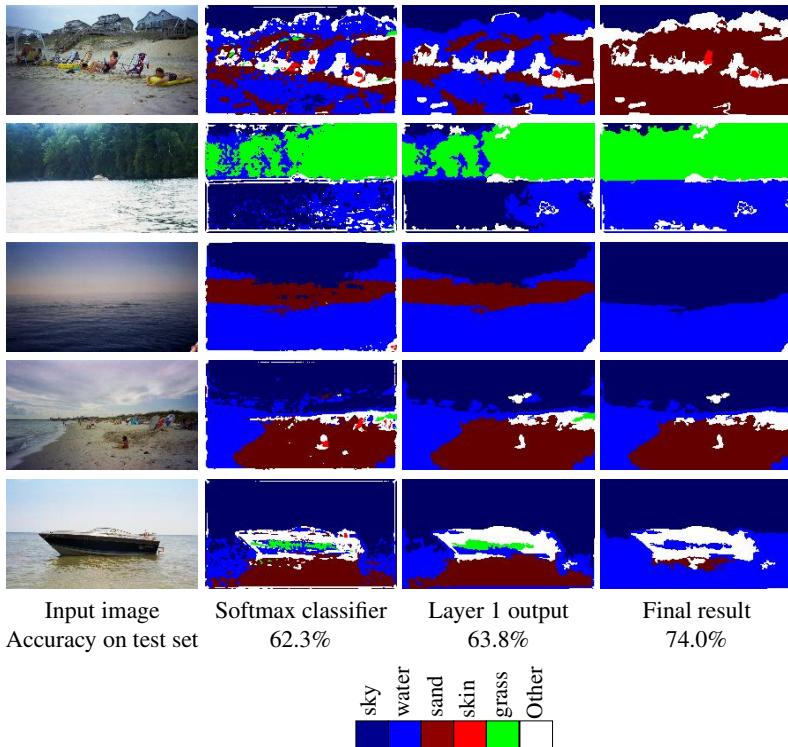
	MRF	Logistic <sup>−</sup>	CRF <sup>−</sup>	Logistic	CRF
DR (%)	58.35	47.50	61.79	60.80	72.54
FP (per image)	2.44	2.28	2.28	1.76	1.76

particularly challenging due to wide within-class variance in the appearance of the data due to drastic illumination conditions. Another characteristic of this dataset which makes it difficult is that, for most of the images, a significant area belongs to none of the semantic classes (i.e., falls under the *other* category). Traditionally it has been hard to model this category because any pixel in this category can virtually have unconstrained appearance.

The layer 1 of hierarchical CRF implements smoothness of pixel labels as the local context. Hence, the sites in layer 1 are image pixels, and three HSV color features and two texture features (based on second moment matrix) give a 5 dimensional unary feature vector. Use of quadratic kernel yielded a 21 dimensional feature vector  $h_i(y)$ . To implement label smoothing, the pairwise feature vector  $\mu_{ij}(y)$  is set to 1, resulting in the Potts model.

The layer 2 encodes interactions among different regions given the beliefs at layer 1 and a partition. Each region of the partition is a site in layer 2. For this dataset, the number of sites at layer 2 varied from 13 to 49 for different images. Each node in this layer is connected to every other node. The computations over these complete graphs are still efficient because of the small number of nodes in the graph. The unary feature vector for each node  $r$  consists of normalized product of beliefs from all the sites  $i$  in  $S_r^{(1)}$  and the normalized centroid location of the region  $r$ . This gives an 8 dimensional feature vector. Further, quadratic transforms are used to obtain a 44 dim vector. The pairwise features between regions are chosen to be binary indicator attributes: a region is *above*, *beside* or *enclosed* within another region.

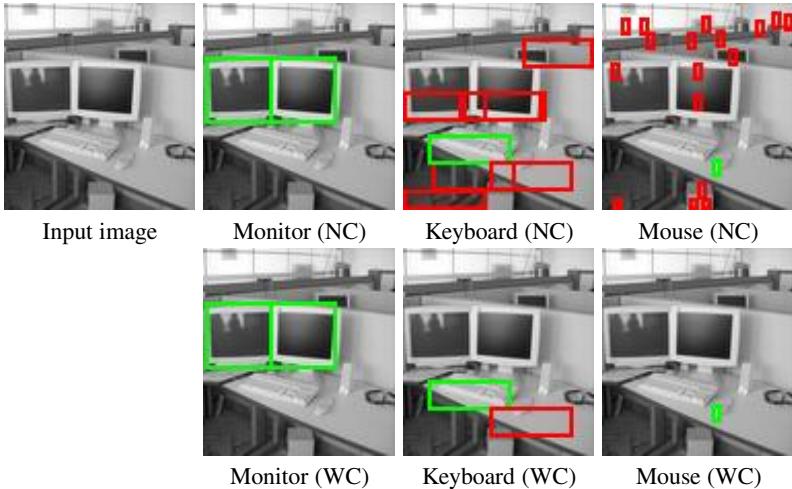
A few example results from the test set are shown in Figure 4.9. The softmax classifier (second column) does not perform well because it classifies each pixel independently without considering interactions in the labels. For example, there is substantial confusion between sand and skin regions or water and sky regions. In addition, the labels are not smooth giving the resulting classification a dithered appearance. The smoothness of labels can be achieved (third column) by implementing smoothing interaction potential in layer 1 of the hierarchical CRF. However, the errors in the larger regions are not eliminated. But, when the full hierarchical model is applied where the second layer enforces the spatial configuration of the regions, most of the errors are eliminated. Note that there are several images that contain pixels which do not belong to any of the semantic classes (e.g., clothing, chairs, boat etc). Such regions are also classified well by the hierarchical CRF. The last row in Figure 4.9 shows that the average accuracy on the test set increases to 74% using the full hierarchical model in comparison to 62.3% from the softmax classifier.



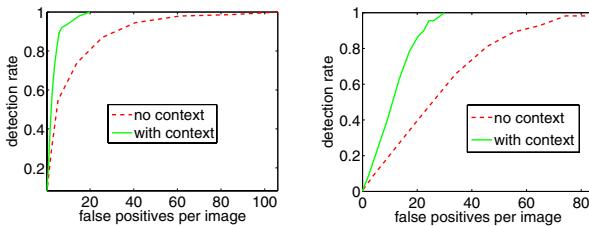
**Fig. 4.9** Pixelwise classification results on the Beach dataset using hierarchical CRFs. 'Layer 1 output' shows the result of implementing label interactions through layer 1 only. Label smoothing is achieved but many large regions are labeled wrong in this output. 'Final result' shows the final classification using both the layers in the hierarchical model which eliminates most of the errors.

The second set of experiments aims to detect objects i.e., monitor, keyboard and mouse in an office scene. The dataset contained 164 low-resolution images of size less than  $100 \times 100$  pixels each [21]. The main challenge in the dataset is the detection of the keyboard and the mouse, which spanned only a few pixels in the images. For these experiments, the hierarchical CRF enforces interactions among the three objects, resulting in a significant reduction in false alarms.

For each object, at first a base detector is trained using gentle-boost. Since the size of the mouse in the input images is very small (on average about  $8 \times 5$  pixels), the boosting based detector could not be trained for the mouse. Instead, a simple template matching based detector is learned. A patch at a location, where the output of any of the three detectors is higher than a threshold, represents a site in  $S^{(1)}$ . The set of sites  $S^{(2)}$  in layer 2 is the same as in layer 1, indicating the trivial partition. The label set for the sites in  $S^{(1)}$  and  $S^{(2)}$  is  $\{\text{monitor}, \text{keyboard}, \text{mouse}, \text{background}\}$ . Since layer 1 uses the output of a standard object detector, interactions among sites take place only at layer 2.



**Fig. 4.10** Detection results for monitor, keyboard and mouse using context based on spatial configuration of objects. NC - No Context, WC - With Context. Monitor detection was good with the base detector itself due to less appearance ambiguity. Note the impoverished appearances of the keyboard and the mouse. Green and red indicate true detections and false alarms respectively.



**Fig. 4.11** The ROC curves for the detection of keyboard (left) and mouse (right). Relatively high false alarm rates for the mouse were due to very small size of mouse (about  $8 \times 5$  pixels) in the input images.

The unary features at layer 2 consist of the score from each detector yielding a 3 dimensional feature vector. The difference of coordinates of the patch centers resulted in a 2 dimensional pairwise feature vector. Each node is connected to every other node in this layer. Figure 4.10 shows a typical result from the test set. It is clear that the false alarms are reduced considerably in comparison to the base detector. The use of context did not change the results for the monitor, since the base detector itself was able to give good performance. This is reasonable because one hopes the context to be more useful when the local appearance of an object is more ambiguous. The ROC curves for the keyboard and the mouse detection are compared with the corresponding base detectors in Figure 4.11. For the mouse detection, even though the hierarchical CRF was able to reduce the false positives significantly, the

number of false alarms per image is still high. This is understandable because the size of mouse is very small in all the images. One can hope for context to improve detection only if there exists at least 'bare-minimum' appearance based evidence for that object in images.

## 4.7 Related Work and Further Readings

In this chapter, we gave a succinct review of basic types of CRFs used in computer vision. One can find a more in-depth discussion on modeling, parameter learning and inference in these CRFs in [9]. It also contains extensive details on the experimental procedures including feature extraction and speed comparisons.

CRFs were introduced in computer vision by Kumar and Hebert [11] [12] extending the 1D-CRFs from Lafferty et al. [15]. Since then, a number of techniques have been proposed in vision that further modified CRFs for various applications. Different types of local classifiers such as neural network [7], boosted stumps [21] and probit function [19] have been used to model clique potentials. A Hidden CRF model was introduced in [20] to handle latent variables. Learning in CRFs was extended to a semi-supervised paradigm by [17]. As a final note, we would like to mention that taking a non-probabilistic view, energy based models have been used in vision. These models have expressive power similar to CRFs [2] [16]. However, effective parameter learning is perhaps the biggest challenge in such non-probabilistic models.

## References

1. Besag, J.: On the statistical analysis of dirty pictures. *Journal of Royal Statistical Society B*-48, 259–302 (1986)
2. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In: *Proceedings of the International Conference on Computer Vision (ICCV)*, vol. I, pp. 105–112 (2001)
3. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. *IEEE transactions on Pattern Analysis and Machine Intelligence* 23(11), 1222–1239 (2001)
4. Geman, S., Geman, D.: Stochastic relaxation, gibbs distribution and the bayesian restoration of images. *IEEE transactions on Pattern Analysis and Machine Intelligence* 6, 721–741 (1984)
5. Greig, D.M., Porteous, B.T., Seheult, A.H.: Exact maximum a posteriori estimation for binary images. *Journal of Royal Statistical Society* 51(2), 271–279 (1989)
6. Hammersley, J.M., Clifford, P.: Markov field on finite graph and lattices (unpublished)
7. He, X., Zemel, R., Carreira-Perpinan, M.: Multiscale Conditional Random Fields for image labelling. In: *Proceedings of International Conference on Computer Vision and Pattern Recognition* (2004)
8. Ising, E.: Beitrag zur theorie der ferromagnetismus. *Zeitschrift Fur. Physik.* 31, 253–258 (1925)

9. Kumar, S.: Models for Learning Spatial Interactions in Natural Images for Context-Based Classification. PhD Thesis, Carnegie Mellon University, The Robotics Institute, School of Computer Science (2005)
10. Kumar, S., August, J., Hebert, M.: Exploiting inference for approximate parameter learning in discriminative fields: An empirical study. In: Rangarajan, A., Vemuri, B.C., Yuille, A.L. (eds.) EMMCVPR 2005. LNCS, vol. 3757, pp. 153–168. Springer, Heidelberg (2005)
11. Kumar, S., Hebert, M.: Discriminative fields for modeling spatial dependencies in natural images. In: Proceedings of the International Conference on Neural Information Processing Systems, NIPS (2003)
12. Kumar, S., Hebert, M.: Discriminative Random Fields: A discriminative framework for contextual interaction in classification. In: Proceedings of the International Conference on Computer Vision (ICCV), vol. 2, pp. 1150–1157 (2003)
13. Kumar, S., Hebert, M.: A hierarchical field framework for unified context-based classification. In: Proceedings of the International Conference on Computer Vision, ICCV (2005)
14. Kumar, S., loui, A.C., Hebert, M.: An observation-constrained generative approach for probabilistic classification of image regions. Image and Vision Computing, Special Issue on Generative Models Based Vision 21, 87–97 (2003)
15. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the International Conference on Machine Learning (2001)
16. LeCun, Y., Huang, F.J.: Loss functions for discriminative training of energy-based models. AI-Stats (2005)
17. Lee, C., Wang, S., Jiao, F., Schuurmans, D., Greiner, R.: Learning to model spatial dependency: semi-supervised discriminative random fields. In: Proceedings of the International Conference on Neural Information Processing Systems Conference, NIPS (2006)
18. McCullagh, P., Nelder, J.A.: Generalised Linear Models. Chapman and Hall, London (1987)
19. Qi, Y., Szummer, M., Minka, T.P.: Diagram structure recognition by Bayesian Conditional Random Fields. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, CVPR (2005)
20. Quattoni, A., Collins, M., Darrell, T.: Conditional Random Fields for object recognition. In: Proceedings of the International Conference on Neural Information Processing Systems, NIPS (2004)
21. Torralba, A., Murphy, K.P., Freeman, W.T.: Contextual models for object detection using Boosted Random Fields. In: Proceedings of the International Conference on Neural Information Processing Systems, NIPS (2005)

# Chapter 5

## From the Subspace Methods to the Mutual Subspace Method

Ken-ichi Maeda

**Abstract.** The *Subspace Method* [25, 21] is a classic method of pattern recognition, and has been applied to various tasks. The *Mutual Subspace Method* [19] is an extension of the *Subspace Methods*, in which *canonical angles* (*principal angles*) between two subspaces are used to define *similarity* between two patterns (or two sets of patterns). The method is applied to face recognition and character recognition in Toshiba Corporation. The *Karhunen-Loëve eigenvalue method* or *Principal Component Analysis (PCA)* [8, 13, 17] is a well-known approach to form a subspace that approximates a distribution of patterns, and it was introduced as a tool of pattern recognition [10, 24]. The extension from the *Subspace Methods* to the *Mutual Subspace Method* corresponds to the difference between *PCA* and *Canonical Correlation Analysis (CCA)* [19]<sup>1</sup>. In this chapter, the *Mutual Subspace Method*, its mathematical foundations and its applications are described.

### 5.1 Introduction

Similarity definition is the most important factor for pattern recognition since no two objects are identical in the real worlds. Similarity can be defined in various ways, but here we take one based on the so-called *pattern matching*. The easiest example is *template matching*, e.g., as shown in an optical character reader (OCR) patent by Tauschek [22] (see Figure 5.1).

A more mathematical interpretation of such a simple matching is as follows: An image is a set of light intensity,  $f(x, y)$ , where  $(x, y)$  is a position. Sometimes we write  $f(\mathbf{r})$ , where  $\mathbf{r} = xi + yj$ . We can appropriately quantise each position and divide whole area into  $I$  meshes. Thus an image is represented with a vector,

---

Ken-ichi Maeda

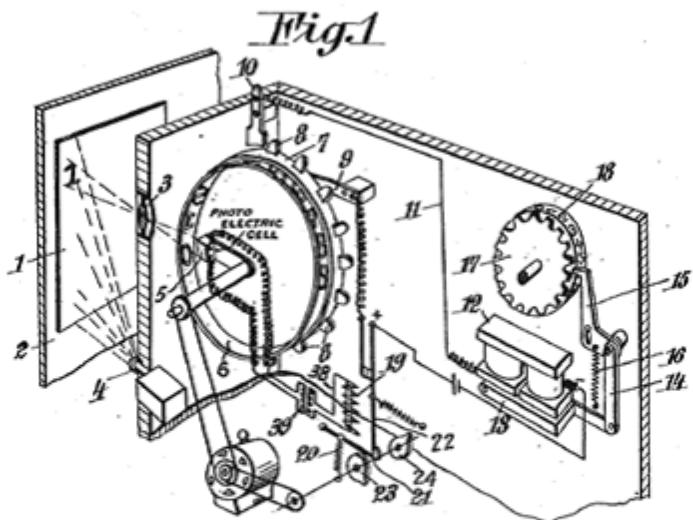
Corporate Research & Development Center, Toshiba Corporation, Japan  
e-mail: [ken.maeda@toshiba.co.jp](mailto:ken.maeda@toshiba.co.jp)

<sup>1</sup> A text book by Bishop [4] is recommended to learn about the difference between *PCA* and *CCA*, as well as to learn about more general statistical background of pattern recognition.

G. TAUSCHEK

READING MACHINE

Filed May 27, 1929

*Fig.2.*

**Fig. 5.1** An example of template matching: Tauschek's OCR patent (US Patent 2,026,329).

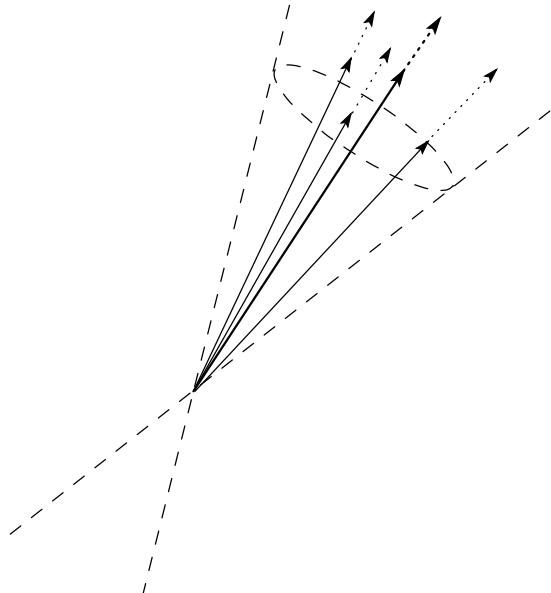
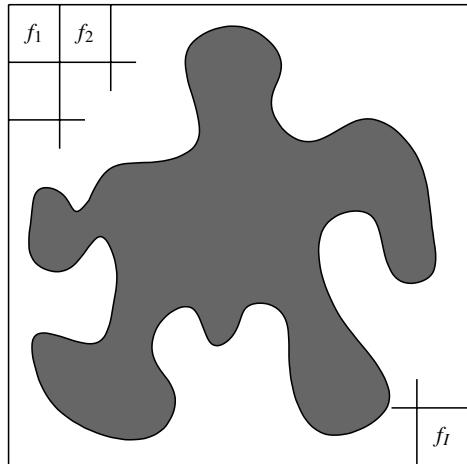
$$\mathbf{f} = (f_1, f_2, \dots, f_i, \dots, f_l)^t, \quad (5.1)$$

where  $f_i$  is the intensity of the  $i$ -th mesh (see Figure 5.2). We call this “vector representation of image pattern (or just pattern).”

There exist no two objects that are identical in the real world, (i.e., some mesh values should be different between any two patterns). Thereby a set of patterns which belong to the same class are distributed in a certain area.

If the illumination becomes  $a$ -times brighter, each mesh value of a pattern becomes  $a$ -times larger. Thus the area has a shape of cone, as shown in Figure 5.3.

**Fig. 5.2** Vector representation of image pattern: each mesh value is light intensity of the position.



**Fig. 5.3** Cone shape pattern distribution in a class.

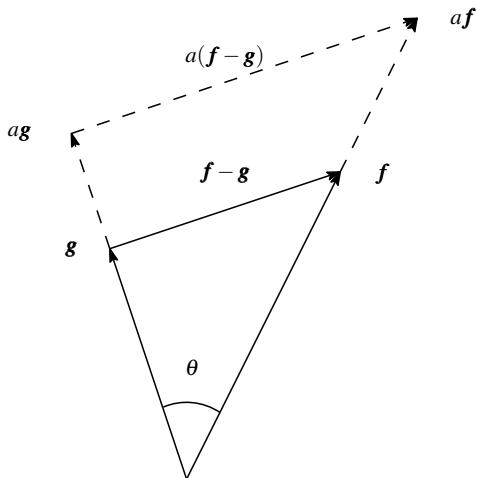
On one hand, the simplest way to define a similarity between two patterns is to use the distance between the two vectors that represent the patterns, as

$$\mathbf{d} = \mathbf{f} - \mathbf{g}. \quad (5.2)$$

However, if the distance is used, it directly reflects the change of illumination; if the illumination becomes  $a$ -times brighter, the distance is also  $a$ -times longer.

On the other hand, another way to define a similarity is to use the angle between the two vectors, as

**Fig. 5.4** Comparison of similarity definitions: distance or angle.



$$\cos^2 \theta = \frac{(\mathbf{f}, \mathbf{g})^2}{(\|\mathbf{f}\|^2 \|\mathbf{g}\|^2)}. \quad (5.3)$$

We call this “the *simple similarity* between  $\mathbf{f}$  and  $\mathbf{g}$ .” A merit of using the angle is that it is unchanged under the illumination change. See Figure 5.4 for comparison of these definitions.

However such a simple matching method has a problem caused by change in shape or position even though the amount of change is small. Even a single bit shift as shown in Figure 5.5 may make a serious change in similarity; the inner product of these two patterns is 0!

We have to introduce a more powerful matching method in order to solve the problem. The *Subspace Methods* and the *Mutual Subspace Method* were invented in order to define similarities which were more stable against pattern variations and more discriminative against similar classes.

-1	-1	-1	-1
+1	+1	+1	+1
+1	+1	+1	+1
-1	-1	-1	-1

-1	-1	-1	-1
-1	-1	-1	-1
+1	+1	+1	+1
+1	+1	+1	+1

**Fig. 5.5** An example of problem with simple matching: A single bit shift makes the inner product value 0.

## 5.2 The Subspace Methods

### 5.2.1 A Brief History of the Subspace Methods

A solution to the problem with *template matching* is to use the *Subspace Methods*<sup>2</sup>. The *Subspace Methods* are a kind of the *pattern matching method*, where a more sophisticated matching than that in *template matching* is employed, i.e., the *dictionary* or the reference pattern is represented with not a single pattern but a subspace. *Karhunen-Loëve expansion (K-L expansion)*, which is employed to make the subspace, was independently introduced by Iijima [10] and Watanabe [24] in pattern recognition at almost the same time<sup>3</sup>. The name of “the *Subspace Method*” was given by Watanabe et al. [25].

Iijima’s method was originally called “the *Multiple Similarity Method* [11][12],” and corresponding Watanabe’s method was called “*CLAFIC* [25].” They are slightly different, but the process are almost the same:

- use a subspace as a dictionary of each class,
- collect a set of known patterns of each class and perform *K-L expansion* to make the dictionary<sup>4</sup> and
- calculate measures for all classes to classify an unknown pattern.

It is interesting that the *K-L method* itself was also independently invented by Karhunen [13] and Loëve [17] for the research of stochastic process at almost the same time. A discrete form was proposed as *Principal Component Analysis (PCA)* by Hotelling [8] earlier than them.

The *K-L method* is occasionally employed also in order to reduce dimensionality; which is called “*K-L transformation*.” Watanabe’s corresponding method is called “*SELFIC* [24].” Almost the same method was revived by Turk and Pentland [23] for face recognition.

The *Subspace Methods of pattern recognition* have been investigated also by many researchers, such as Fu and Yu [5], Kittler and Young [15] and Kohonen [16]. There are a number of variations in the *Subspace Methods*, such as “*MOSS*,” “the *Learning Subspace Method*,” “the *Orthogonal Subspace Methods*,” etc. For further detail, refer to Oja’s text book. For those interested in history, refer to Grenander’s book [7].

### 5.2.2 Basic Idea

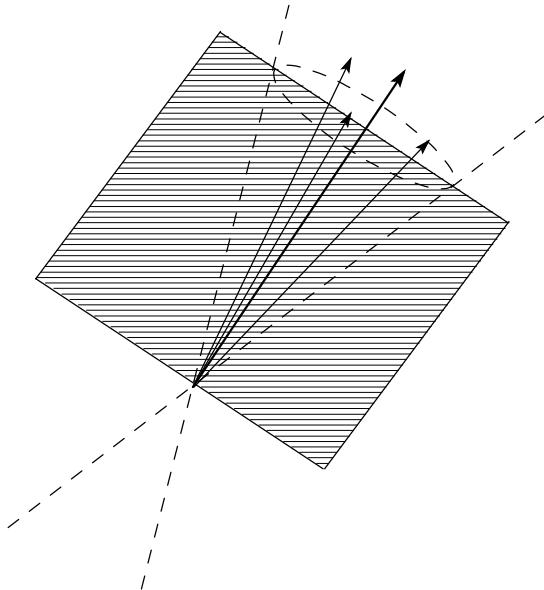
As discussed before, no two patterns are identical in the real world. The question is what should be the representative of a class, or the reference pattern: We call it “a *dictionary* of the class.” The simplest idea is to assign the mean vector as the dictionary. An alternative is to use all sample vectors. The former is too simple and the

<sup>2</sup> A good text book [21] is unfortunately out of print, but it may be found in a library.

<sup>3</sup> Iijima called it “*Mode Function Expansion* [10].”

<sup>4</sup> Iijima et al. [11] also showed another way to make a subspace using derivation.

**Fig. 5.6** Pattern distribution approximation by a subspace.



latter requires a large memory to store all vectors. Viewing the pattern distribution in a class shown in Figure 5.3, we find the cut plane is an ellipse. The ratio of the major axes and the minor axes of the ellipse is usually large.

If the pattern distribution has such a large ratio of ellipse shape, the distribution can be approximated by a plane (i.e., a subspace). An example of the approximation is shown in Figure 5.6. Instead of calculating the angle between an unknown input pattern and all known sample patterns, we can define a similarity as the angle between the unknown pattern and the subspace, as shown in Figure 5.7 and Equation (5.5).

$$S_{MS}^{(l)}[\mathbf{f}] \stackrel{\text{def}}{=} \cos^2 \theta \quad (5.4)$$

$$= \sum_{m=1}^M \frac{(\mathbf{f}, \boldsymbol{\varphi}_m)^2}{\|\mathbf{f}\|^2}. \quad (5.5)$$

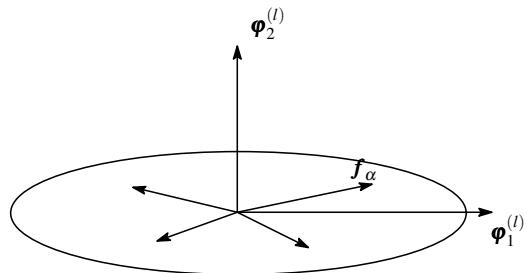
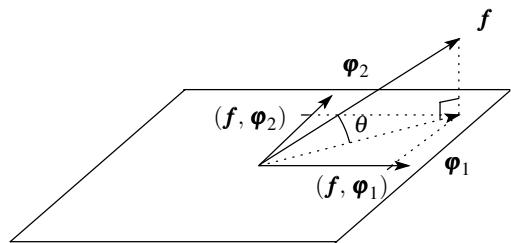
where  $\mathbf{f}$  is an unknown input pattern and  $\{\boldsymbol{\varphi}_m^{(l)}\}_{m=1}^M$  are an orthonormal basis of a subspace for class ' $l$ '. We call this "the *multiple similarity* of  $\mathbf{f}$  for class ' $l$ '."

The angle between an unknown input pattern and a dictionary subspace is the angle between the unknown pattern and the nearest vector in the subspace. Thereby a subspace dictionary is equivalent to preparing infinite number of sample patterns.

### 5.2.3 Subspace Construction

Now the problem is how to make the subspace, more practically how to find the orthonormal basis of the subspace. We follow Iijima's idea [10] for this purpose.

**Fig. 5.7** Similarity definition by the angle between a pattern and a subspace.



**Fig. 5.8** Cut plane of the pattern distribution of class ‘*l*’.

Figure 5.8 shows the cut plane of the pattern distribution of class ‘*l*.’ We may think the figure shows an orthogonal projection of the pattern distribution to the complementary subspace that is orthogonal to the standard pattern. The problem is interpreted as how to find  $\{\boldsymbol{\varphi}_m^{(l)}\}_{m=1}^M$ .

Let  $\{f_\alpha\}$  and  $\{w_\alpha^{(l)}\}$  be a set of sample patterns and their probability of existence in a class ‘*l*,’ respectively. Then the mean value of the *simple similarities* between a pattern  $f$  and the sample patterns  $f_\alpha$  is

$$S^{(l)}[f] = \sum_{\alpha} w_\alpha^{(l)} \frac{(f, f_\alpha)^2}{\|f\|^2 \|f_\alpha\|^2} \quad (5.6)$$

$$= \sum_{m=1}^M \frac{\lambda_m^{(l)} (f, \boldsymbol{\varphi}_m^{(l)})^2}{\|f\|^2}. \quad (5.7)$$

where  $\{\lambda_m^{(l)}\}_{m=1}^M$  and  $\{\boldsymbol{\varphi}_m^{(l)}\}_{m=1}^M$  are eigenvalues and eigenvectors of the following  $K^{(l)}$ , respectively.

$$K^{(l)} = \sum_{\alpha} w_\alpha^{(l)} \frac{f_\alpha, f_\alpha}{\|f_\alpha\|^2} \quad (5.8)$$

$$= \sum_{m=1}^M \lambda_m^{(l)} \langle \boldsymbol{\varphi}_m^{(l)}, \boldsymbol{\varphi}_m^{(l)} \rangle. \quad (5.9)$$

where  $\langle \bullet, \bullet \rangle$  denotes dyad or Neumann-Schatten product.

Now the problem is to find the  $\boldsymbol{\varphi}$  that maximises  $S^{(l)}[\boldsymbol{\varphi}]$ . The  $\boldsymbol{\varphi}$  should satisfy  $\delta S^{(l)} = 0$ . Let  $\mathbf{p}_\alpha$  be  $f_\alpha / \|f_\alpha\|$ . Then Equation (5.6) is

$$S^{(l)}[\boldsymbol{\varphi}] = \sum_{\alpha} w_{\alpha}^{(l)} \frac{(\boldsymbol{\varphi}, \mathbf{p}_{\alpha})^2}{\|\boldsymbol{\varphi}\|^2}. \quad (5.10)$$

Calculating variation of Equation (5.10),

$$\|\boldsymbol{\varphi}\|^2 \delta S^{(l)} + 2S^{(l)}(\boldsymbol{\varphi}, \delta \boldsymbol{\varphi}) = 2 \sum_{\alpha} w_{\alpha}^{(l)} (\boldsymbol{\varphi}, \mathbf{p}_{\alpha})(\mathbf{p}_{\alpha}, \delta \boldsymbol{\varphi}) \quad (5.11)$$

$$= 2(K^{(l)} \boldsymbol{\varphi}, \delta \boldsymbol{\varphi}). \quad (5.12)$$

Since local maxima should satisfy  $\delta S^{(l)} = 0$ ,

$$(S^{(l)} \boldsymbol{\varphi}, \delta \boldsymbol{\varphi}) = (K^{(l)} \boldsymbol{\varphi}, \delta \boldsymbol{\varphi}), \quad (5.13)$$

for any  $\delta \boldsymbol{\varphi}$ . Thus we have an eigenvalue problem,

$$S^{(l)} \boldsymbol{\varphi} = K^{(l)} \boldsymbol{\varphi}, \quad (5.14)$$

which is known as the *Karhunen-Loëve eigenvalue method* or *PCA*.

## 5.3 The Mutual Subspace Method

### 5.3.1 Basic Idea

The reference patterns are represented with subspaces in the *Subspace Methods*. Thinking about the difference between the *simple similarity* and the *multiple similarity*, it is dissymmetric to replace just one of two vectors by a subspace. What about representing both the input and reference patterns with subspaces?

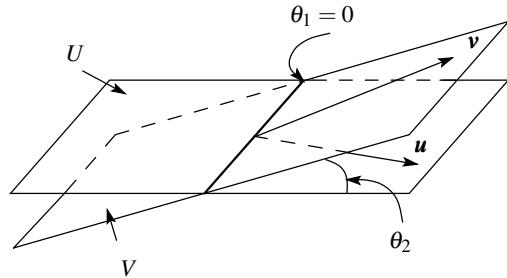
The use of subspaces in the *Subspace Methods* improved the stability of similarity against the changes of pattern positions or shapes. If both the input and reference patterns are represented with the subspaces, we may expect more stability against the changes.

Before extending the definition of similarity from the angle between a vector and a subspace to that between two subspaces, we should define what the angle between two subspaces is. Given the subspaces,  $U$  and  $V$ , the angle between these is defined as the minimum angle between vectors  $\mathbf{u}$  and  $\mathbf{v}$ , where  $\mathbf{u} \in U$  and  $\mathbf{v} \in V$ , according to Dixmier [4]. See Figure 5.9 for a more concrete image. Let  $\theta$  be the angle. Then

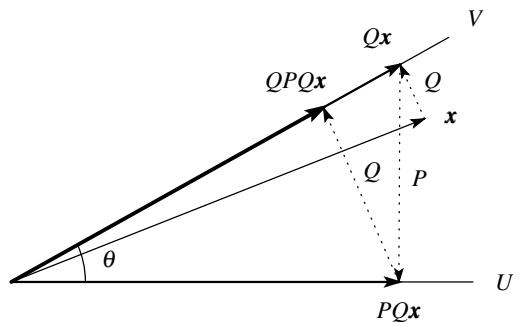
$$\cos^2 \theta \stackrel{\text{def}}{=} \sup_{\substack{\mathbf{u} \in U, \mathbf{v} \in V \\ \|\mathbf{u}\| \neq 0, \|\mathbf{v}\| \neq 0}} \frac{|(\mathbf{u}, \mathbf{v})|^2}{\|\mathbf{u}\|^2 \|\mathbf{v}\|^2}. \quad (5.15)$$

We use this value also as the definition of similarity. For actual calculation of the similarity, we apply the following theorem.

**Fig. 5.9** Angle between two subspaces: Compare to angle between a vector and a subspace shown in Fig. 5.7



**Fig. 5.10** How to calculate the angle between two subspaces: The length of  $QPQx$  is  $\cos^2 \theta$  times the length of  $Qx$ .



### Theorem 1

Let  $U$  and  $V$  be two subspaces and  $P$  and  $Q$  be orthogonal projection operators onto  $U$  and  $V$ , respectively. Then the angle between  $U$  and  $V$  is calculated as the maximum eigenvalue of  $PQP$  or  $QPQ$  [19][20][3]. Let  $\mu$  and  $\nu$  be the maximum eigenvalues of  $PQP$  and  $QPQ$ , i.e.,

$$PQPx = \mu x, \quad (5.16)$$

and

$$QPQx = \nu x, \quad (5.17)$$

respectively. Then

$$\cos^2 \theta = \|QP\|^2 \quad (5.18)$$

$$= \|PQ\|^2 \quad (5.19)$$

$$= \mu \quad (5.20)$$

$$= \nu, \quad (5.21)$$

where the norm of an operator  $A$  is defined as

$$\|A\| \stackrel{\text{def}}{=} \sup_{\|\mathbf{z}\| \neq 0} \frac{\|A\mathbf{z}\|}{\|\mathbf{z}\|}. \quad (5.22)$$

(see Figure 5.10 for intuitive understanding).

According to this theorem, we may use the eigenvalue,  $\mu$  or  $\nu$ , as the similarity instead of the angle,  $\cos^2 \theta$ . However, as the eigenvalue calculation of  $PQP$  or  $QPQ$  is costly due to the large size of the matrices, we practically translate the eigenvalue problem into that of a smaller matrix  $X$  whose eigenvalues are identical to  $PQP$  or  $QPQ$ ,

$$X \mathbf{z} = \lambda \mathbf{z}, \quad (5.23)$$

$$X = \begin{pmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1n} \\ \vdots & & \vdots & & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{in} \\ \vdots & & \vdots & & \vdots \\ x_{m1} & \dots & x_{mj} & \dots & x_{mn} \end{pmatrix}, \quad (5.24)$$

where

$$x_{ij} = \sum_{m=1}^M (\boldsymbol{\psi}_i, \boldsymbol{\varphi}_m) (\boldsymbol{\varphi}_m, \boldsymbol{\psi}_j), \quad (5.25)$$

or

$$x_{ij} = \sum_{n=1}^N (\boldsymbol{\varphi}_i, \boldsymbol{\psi}_n) (\boldsymbol{\psi}_n, \boldsymbol{\varphi}_j), \quad (5.26)$$

where  $\{\boldsymbol{\varphi}_m\}_{m=1}^M$  are a set of basis of  $U$  whose dimension is  $M$ , and  $\{\boldsymbol{\psi}_n\}_{n=1}^N$  are of  $V$  whose dimension is  $N$ .

### 5.3.2 Application to Chinese Character Recognition

The *Mutual Subspace Method* was originally developed for Chinese Character (Kanji) recognition, in particular hand-printed Kanji recognition. Examples of hand-printing are shown in Figure 5.11.

The problems of hand-printed Kanji recognition are:

- many classes (more than 1,000, maybe 200,000),
- hand-printing variations (such as shown in Figure 5.11) and
- existence of similar classes (such as shown in Figure 5.11).

The existence of many classes makes it difficult to collect the enough number of sample patterns for making the dictionaries using the *K-L method*<sup>5</sup>. The *Mutual Subspace Method* was mainly a solution to this problem; we expected a more robust similarity against the variations without collecting so many patterns.

The next step is to make input subspace. If we assume that the major variations of patterns are only position shift in  $x$  and  $y$  directions and that the amount of shift is small, we can approximate the variations using derivations in  $x$  and  $y$  directions. We

---

<sup>5</sup> We empirically know that the number of required training patterns is the cube of the subspace dimension. If we make 1,000 10-dimensional subspaces, we need 1,000,000 patterns for training.



**Fig. 5.11** Examples of hand-printed Kanji: 3 different classes that mean ‘know,’ ‘weave’ and ‘fiber.’

can make a 3-dimensional subspace using original and the differentiated patterns in  $x$  and  $y$  directions.<sup>6</sup> Thus we can make a subspace without the *K-L method*. We call it “the *differential method of dictionary*.”

It is a good point to think of a *blurring filter*. The *blurring filter* is effective to make matching easier because it reduces high frequency noises. The most common blurring filter is made with the *Gaussian function*. Let  $G(\mathbf{r}, \sigma)$  be a *Gaussian function*,

$$G(\mathbf{r}, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|\mathbf{r}\|^2}{2\sigma^2}\right), \quad (5.27)$$

and  $f(\mathbf{r})$  be a pattern. Then the *blurring process* is as follows,

$$f(\mathbf{r}_i, \sigma) = \int G(\mathbf{r}_i - \mathbf{r}, \sigma) f(\mathbf{r}) d\mathbf{r}. \quad (5.28)$$

However, since there are mutually similar classes as discussed above, simple blurring may cause degradation of the recognition accuracy. In order to intensify small differences among the mutually similar classes, Maeda et al. [18] introduced order  $m$  derivation in  $x$  direction and order  $n$  in  $y$  direction,

---

<sup>6</sup> We need the Gram-Schmidt procedure in order to make the orthonormal basis of the subspace.

$$\frac{\partial^{m+n}}{\partial x^m \partial y^n} f(\mathbf{r}, \sigma) = \int \frac{\partial^{m+n}}{\partial x'^m \partial y'^n} G(\mathbf{r} - \mathbf{r}', \sigma) f(\mathbf{r}') d\mathbf{r}' \quad (5.29)$$

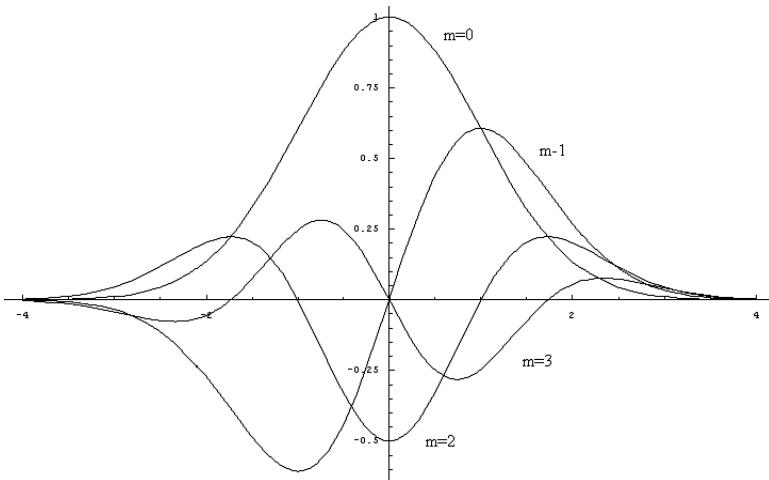
$$= \frac{\sqrt{m!n!}}{(-\sqrt{2})^{m+n}} f_{mn}(\mathbf{r}, \sigma), \quad (5.30)$$

where

$$f_{mn}(\mathbf{r}, \sigma) = \int \frac{1}{\sqrt{m!n!} \left( \frac{\sigma}{\sqrt{2}} \right)^{m+n}} G(\mathbf{r} - \mathbf{r}', \sigma) H_m\left(\frac{x-x'}{\sigma}\right) H_n\left(\frac{y-y'}{\sigma}\right) f(\mathbf{r}') d\mathbf{r}'. \quad (5.31)$$

$H_m$  is the *Hermite Polynomial* of order  $m$ . – Isn't this process the same as the *differential method* of dictionary?

The integral kernel of Equation (5.31) is the *Gaussian weighted Hermite Polynomials*. We call this “the *Gauss-Hermite Kernel*.” The 1-dimensional shapes are shown in Figure 5.12.



**Fig. 5.12** 1-dimensional shapes of the *Gauss-Hermite Kernel*: The shapes resemble the *Gabor Functions*.

Since *Hermite Polynomials* are mutually orthogonal with a Gaussian weight, we can have a set of orthonormal basis of a subspace which represents an input pattern<sup>7</sup>.

The experimental results on hand-printed Kanji recognition, just with 500 patterns for each class shown in Figure 5.11, using the *simple similarity* (SS), the *Subspace Method* (SM) and the *Mutual Subspace Method* (MSM) showed the effectiveness of the proposed method (see Table 5.1).

<sup>7</sup> In most cases, we also need the *Gram-Schmidt* procedure since we usually define the inner product with no weight.

**Table 5.1** Hand-printed Kanji recognition result for the three mutually similar classes: 500 patterns each.

Method	SS	SM	MSM
Recognition Rate	75.9 %	80.2 %	82.0 %

In *SS* and *MSM* only the mean pattern of a class was used to make the dictionary of the class whereas all patterns were used in *SM*. In *MSM* the  $x$  and  $y$  derivatives of the mean pattern are also used to construct a subspace instead of *K-L expansion*. Even though only the mean pattern was used in the *MSM*, its recognition rate was the best.

### 5.3.3 Application to Face Recognition

Face recognition is an easy-to-use authentication method. A problem of this method is its relatively low accuracy in comparison with other biometric methods, such as finger print recognition, iris recognition and vein recognition. However it still has several advantages over others, such as remote sensing, unrealisedness and many objects at a time.

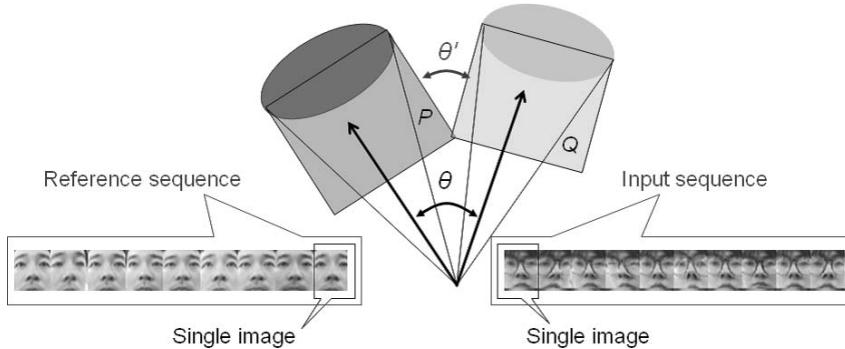
Difficulties in face recognition are mainly caused by

- change of face directions,
- non-rigid objects,
- aging of objects and
- change of illuminations.

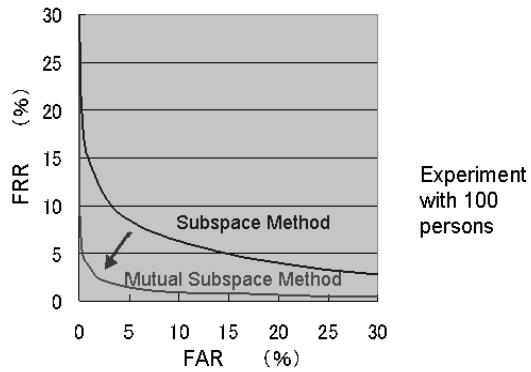
The first two items make relatively short-term variation whereas the last two are long-term.

We focus on the first two problems and show a way to overcome them: The most important point is how to define a similarity that is robust against the variation of input pattern. The input patterns vary according to a probabilistic distribution caused by change of face directions and facial expressions. If we use the moving picture, or an image sequence, as the input instead of a single photograph, we can make a subspace, which approximates the distribution. Thus we can apply the *Mutual Subspace Method*, with which we may expect a more stable similarity against the changes of facial directions and facial expressions. Figure 5.13 shows the conceptual scheme of the *Mutual Subspace Method* applied to face recognition.

We made experiments with 100 persons using the *Subspace Method* and the *Mutual Subspace Method*. The experimental results are shown in the ROC curves in Figure 5.14. The  $x$ -axis is the false acceptance rate (FAR) and the  $y$ -axis is the false rejection rate (FRR). The equal error rate (EER) of the *Mutual Subspace Method* was reduced to 1/4 of that of the *Subspace Method*.



**Fig. 5.13** Conceptual scheme of the *Mutual Subspace Method* applied to face recognition: The similarity is defined as  $\cos^2 \theta'$ , where  $\theta'$  is the canonical angle of two subspaces, instead of  $\cos^2 \theta$ , where  $\theta$  is the angle between two single images.



**Fig. 5.14** Experimental results of face recognition using the *Subspace Method* and the *Mutual Subspace Method*: FAR-FRR ROC curves. EER was reduced to  $1/4$ .

### 5.3.4 Application to 3-D Face Recognition

Another problem of face recognition is potential deception using photograph. The *Mutual Subspace Method* discussed above uses only the smallest canonical angle to define a similarity. Since there is no remarkable difference between a frontal face and a photograph of the same face, their vectors are almost identical; which means even if moving pictures are used, the smallest canonical angle between photograph input and face dictionary of the same person is almost 0. Figures 5.15 and 5.16 show the picture sequences of a moving face and a moving photograph, respectively. The leftmost of the face picture sequence is the frontal face, which is almost the same as that of the photograph.

A more mathematical discussion is as follows: Let  $U$  and  $V$  be the subspaces representing a face and its photograph as shown in Figure 5.9. Then the both frontal patterns are on the line of the intersection of the two subspaces, so the angle between them are almost 0. However, in this case, there is another angle,  $\theta_2$ . We expect that

**Fig. 5.15** Picture sequence of a moving face.



**Fig. 5.16** Picture sequence of a moving photograph.



the angle reflects the difference between face, which is a 3-dimensional (3-D) object, and its photograph, which is 2-dimensional (2-D).

Maeda et al. [19] proposed to use the third smallest angle. Table 5.2 shows the result of a pilot study, in which only the similarities defined with the smallest and the third smallest canonical angles were measured and the reference pattern was made only for the person, P0.

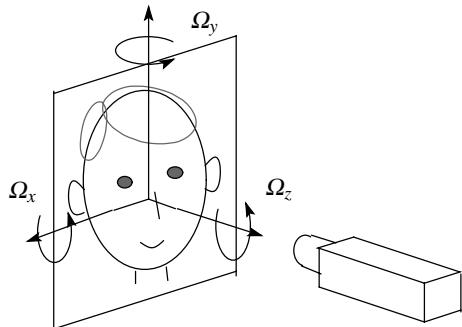
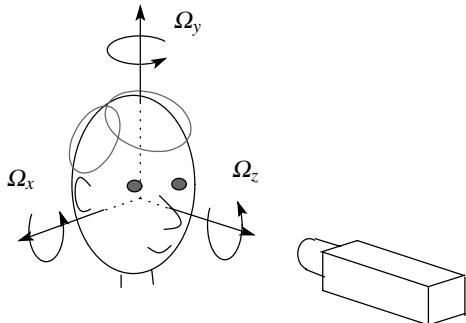
**Table 5.2** Similarity using the smallest and the 3rd smallest canonical angles (modified from Maeda et al. [19]).

Person ID	Smallest CA		3rd smallest CA	
	Face	Photo	Face	Photo
P0	0.989	0.977	0.937	0.204
P1	0.702	0.591	0.256	0.165
P2	0.707	0.619	0.520	0.237
P3	0.786	0.741	0.488	0.123
P4	0.701	0.665	0.457	0.075
P5	0.643	0.626	0.459	0.124
P6	0.730	0.612	0.227	0.055
P7	0.554	0.678	0.334	0.238
P8	0.750	0.732	0.557	0.246
P9	0.716	0.600	0.545	0.154
P10	0.772	0.648	0.435	0.075

According to the result, the similarity using the smallest canonical angle does not have capability to differentiate face and its photograph because the similarity value for the photograph is 0.977 where as that for the face is 0.989; the difference is quite small. However the similarity values using the third smallest canonical angle have enough difference for this purpose; 0.204 and 0.937, respectively.

Now the problem is to determine the angle or angles that is the best for differentiating a face and its photograph with enough authentication accuracy. We first should find the dimension of the subspace that approximates an input pattern distribution. Assuming that a face is a rigid object and that we employ some appropriate position and size normalisation, the major variations of the input are caused by face rotation.

Figures 5.17 and 5.18 show rotations of a face and a face photograph. Among these rotations, we focus on  $\Omega_x$  and  $\Omega_y$  because  $\Omega_z$  is not useful for differentiating

**Fig. 5.17** Rotation of a face.**Fig. 5.18** Rotation of a face photograph.

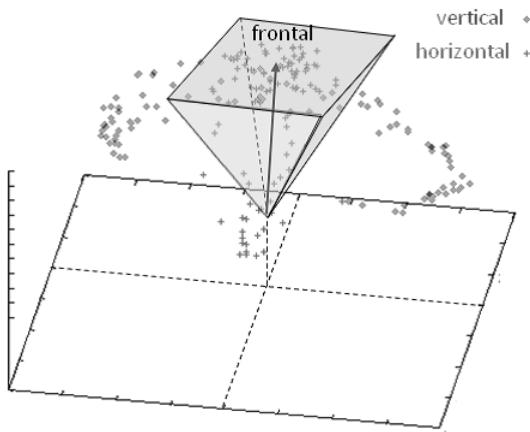
a face and a photograph. Consequently we may suppose that the dimension of the subspace is 3; frontal face, its vertical rotation and horizontal rotation.<sup>8</sup>

Figure 5.19 shows the shape of the manifold (depicted by  $\diamond$  and  $+$ ), on which rotated face vectors exist, and the approximation with a 3-D pyramid, which leads to a 3-D subspace. For intuitive understanding, Figure 5.20 shows the case we consider only vertical rotation. In this case the approximation is a 2-D subspace, and it appears more like a *subspace*. Now that we understand the variation distribution of the face input is approximated with a 3-dimensional subspace, we still wonder if the third smallest angle is the best; what about using the second or the fourth smallest, what about using the angles in combination instead of the single angle on its own? Maeda et al. [20] made experiments to find the difference between the second and the third smallest angles, using convex curved photographs (see Figure 5.21).

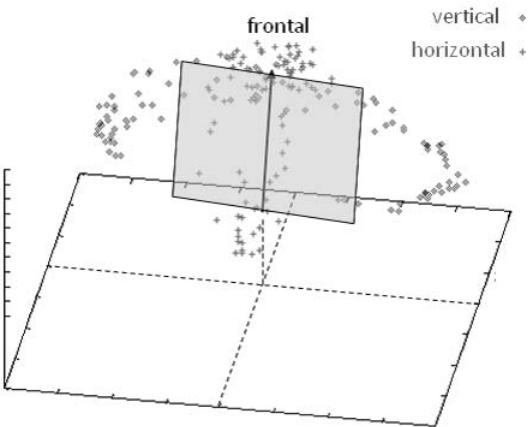
A convex curved photograph appears more like the actual face than a flat photograph, and it is an easy means if it works for deception. Also the mean and the product of the similarities made with up to the third smallest angle and the product of the similarities were used in the experiments.

Figures 5.22, 5.23, 5.24 and 5.25 show the FRR and FAR curves using the similarities defined as the  $\cos^2$  of the smallest, the second smallest, the third smallest and the fourth smallest angles on their own. Figures 5.26 and 5.27 show the results

<sup>8</sup> We also assume the amount of rotation is small, so that the linear approximation can be applied.



**Fig. 5.19** The manifold of vertical and horizontal face rotation and its approximation with a 3-D pyramid.

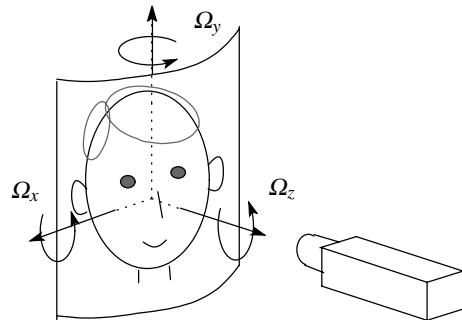


**Fig. 5.20** The approximation with a 2-D subspace considering only vertical rotation.

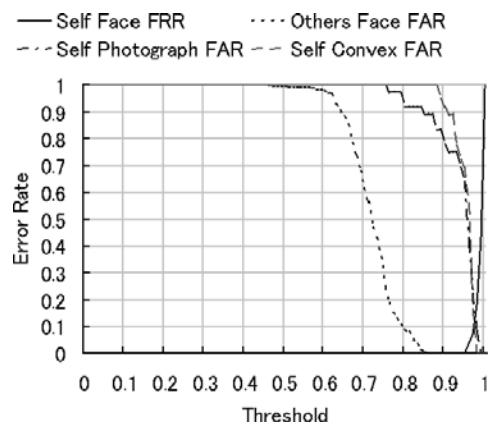
using the mean of the similarities up to the third and the product of the similarities up to the third.

We can confirm that the smallest canonical angle is not effective to differentiate a face and its photograph since the Self FRR and Self Photograph FAR curves cross (see Figure 5.22). Also the Self FRR and Self Convex FAR curves cross. The fourth smallest canonical angle is not effective to identify a face since the Self Face FRR and Others Face FAR curves cross (see Figure 5.25). As a result, we should use the second and/or the third smallest canonical angle, at least.

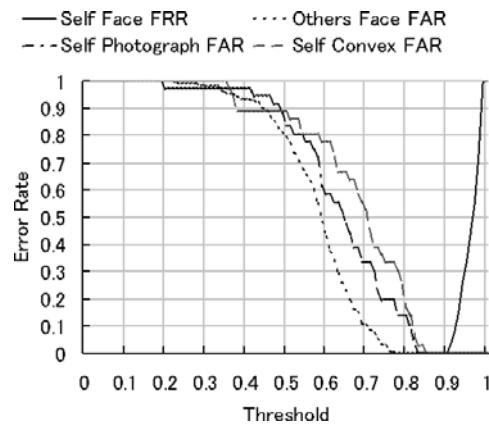
**Fig. 5.21** Rotation of a convex curved photograph.



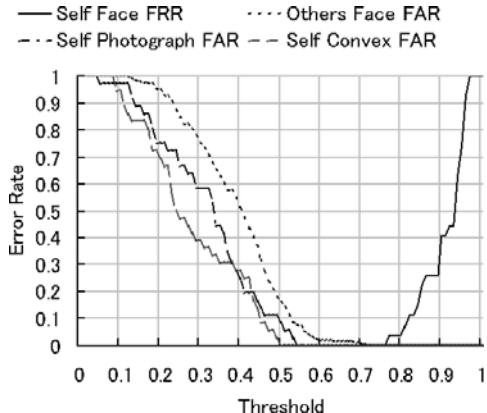
**Fig. 5.22** FRR and FAR using the smallest canonical angle.



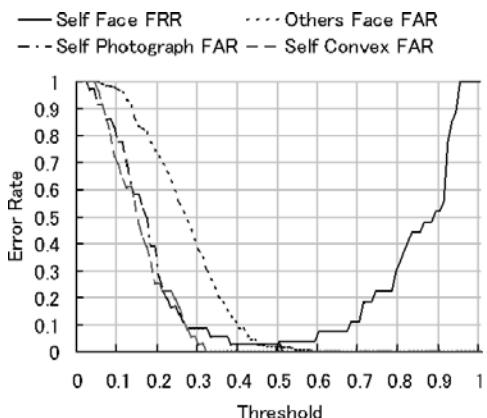
**Fig. 5.23** FRR and FAR using the 2nd smallest canonical angle.



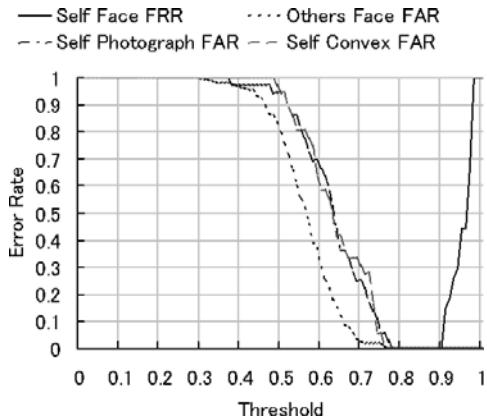
**Fig. 5.24** FRR and FAR using the 3rd smallest canonical angle.



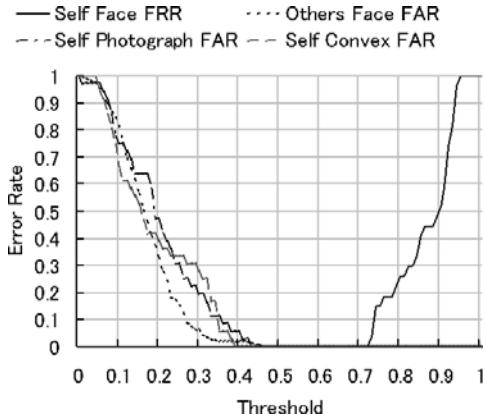
**Fig. 5.25** FRR and FAR using the 4th smallest canonical angle.



**Fig. 5.26** FRR and FAR using the mean of the similarities.



**Fig. 5.27** FRR and FAR using the product of the similarities.



Comparing Figures 5.23 and 5.24, the second smallest canonical angle is better at differentiating a person and others whereas the third is better at differentiating a face and its photograph, in particular for convex curved photograph; there is a trade-off between authentication and anti-deception.

The best one among all of the results is the product of the similarities up to the third canonical angle since the gap between the Self FRR and FARs are the widest among all (see Figure 5.27).

## 5.4 Conclusion

We have discussed a series of pattern matching methods that are based on subspaces. The methods have a long history started with mathematics and physics; the origins were 1930s and 1940s. Even after the *Subspace Methods* were introduced to pattern recognition, more than 40 years have already passed. Since the basic ideas were quite reasonable, the methods have survived for such a long time and have evolved against increasing difficulties in pattern recognition tasks. The *Mutual Subspace Method* is one of such evolutions.

We had to introduce some new mechanisms to overcome the difficulties in more complex recognition objects. For instance, we introduced a new feature extraction based on the *Gauss-Hermite Kernel* for hand-printed Kanji recognition, and we introduced multiple canonical angles for 3-D face recognition. By virtue of the newly introduced mechanisms, the subspace based matching methods have effectively worked and have achieved their aims. However, through the evolution, several basic concepts are common, such as assuming cone shape distributions, using subspaces and using angles for similarity definitions, etc.

We started this chapter with quickly following the history and viewed the evolution from the original *Subspace Method* to the *Mutual Subspace Method*. The evolution has not yet reached the final stage, and hopefully will continue. There

already exist further extensions of the *Mutual Subspace Method*, such as the *Constrained Mutual Subspace Method* [6] and *Tensor Canonical Correlation* [14]. We expect further evolution for more difficult applications in the future.

## References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
2. Björck, Å., Golub, G.H.: Numerical Methods for Computing Angles between Linear Subspaces. Mathematics of Computation 27, 579–594 (1975)
3. Chatelin, F.: Valeurs Propres de Matrices, Masson, Paris (1988)
4. Dixmier, M.: Etude sur les Varietes et les Operatorms de Julia, avec Quelques Applications. Bull. Soc. Math. France 77, 11–101 (1949)
5. Fu, K.S., Yu, T.S.: Statistical Pattern Classification Using Contextual Information. John Wiley and Sons, New York (1980)
6. Fukui, K., Yamaguchi, O.: Face Recognition using multi-viewpoint patterns for robot vision. In: Proceedings of the 11th International Symposium of Robotics Research (ISRR 2003), pp. 192–201 (2003)
7. Grenander, U.: Abstract Inference. John Wiley, New York (1981)
8. Hotelling, H.: Analysis of a Complex Statistical Variables into Principal Components. J. Educ. Psych. 24, 417–441, 498–520 (1933)
9. Hotelling, H.: Relations between Two Sets of Variates. Biometrika 28, 321–377 (1936)
10. Iijima, T.: Basic Theory on Feature Extraction of Visual Pattern. J. IEICE 46, 1714 (1963)
11. Iijima, T., Genchi, H., Mori, K.: A Theoretical Study of Pattern Recognition by Matching Method. In: Proceedings of the First USA-Japan Computer Conference, pp. 42–48 (1972)
12. Iijima, T., Genchi, H., Mori, K.: A Theory of Character Recognition by Matching Method. In: Proceedings of the 1st International Conference on Pattern Recognition, pp. 50–56 (1973)
13. Karhunen, K.: Zur Spektraltheorie stochastischer Prozesse. Ann. Acad. Sci. Fenniae 34 (1946)
14. Kim, T., Wong, S., Cipolla, R.: Tensor Canonical Correlation Analysis for Action Classification. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
15. Kittler, J., Young, P.C.: A New Approach to Feature Selection Based on the Karhunen-Loeve Expansion. Pattern Recognition 5(4), 335–352 (1973)
16. Kohonen, T.: Associative Memory – a System-Theoretical Approach. Springer, Heidelberg (1977)
17. Loèvè, M.: Functions aléatoires du second ordre. Processus stochastique et mouvement Brownien (Lèvy, P.), 366–420, Gauthier-Villars, Paris (1948)
18. Maeda, K., Kurosawa, Y., Asada, H., Sakai, K., Watanabe, S.: Handprinted Kanji Recognition by Pattern Matching Method. In: Proceedings of the International Conference on Pattern Recognition, pp. 789–792 (1982)
19. Maeda, K., Yamaguchi, O., Fukui, K.: Towards 3-Dimensional Pattern Recognition. In: Fred, A., Caelli, T.M., Duin, R.P.W., Campilho, A.C., de Ridder, D. (eds.) SSPR&SPR 2004. LNCS, vol. 3138, pp. 1061–1068. Springer, Heidelberg (2004)
20. Maeda, K., Yamaguchi, O., Fukui, K.: A Fundamental Discussion on 3-Dimensional Pattern Recognition Using Canonical Angles between Subspaces for the Purpose of Differentiating Face and its Photograph. Systems and Computers in Japan 38, 11–20 (2007)

21. Oja, E.: Subspace Methods of Pattern Recognition. Research Studies Press, Letchworth (1983)
22. Tauschek, G.: Reading Machine, US Patent 2,026,329 (1935)
23. Turk, M., Pentland, A.: Face Recognition Using Eigenfaces. In Proceedings of the International Conference on Computer Vision and Pattern Recognition, 453–458 (1993)
24. Watanabe, S.: Karhunen-Loëve Expansion and Factor Analysis. In: Trans. 4th Prague Conf. on Inf. Theory, Stat. Decision Functions, and Random Proc., pp. 635–660 (1965)
25. Watanabe, S., Lambert, P.F., Kulikowsky, C.A., Buxton, J.L., Walker, R.: Evaluation and Selection of Variables in Pattern Recognition. In: Tou, J. (ed.) Computer and Information Science II, pp. 91–122. Academic Press, New York (1967)

## Chapter 6

# What, Where and Who? Telling the Story of an Image by Activity Classification, Scene Recognition and Object Categorization

Li Fei-Fei and Li-Jia Li

**Abstract.** We live in a richly visual world. More than one third of the entire human brain is involved in visual processing and understanding. Psychologists have shown that the human visual system is particularly efficient and effective in perceiving high-level meanings in cluttered real-world scenes, such as objects, scene classes, activities and the stories in the images. In this chapter, we discuss a generative model approach for classifying complex human activities (such as croquet game, snowboarding, etc.) given a single static image. We observe that object recognition in the scene as well as scene environment classification of the image facilitate each other in the overall activity recognition task. We formulate this observation in a graphical model representation where activity classification is achieved by combining information from both the object recognition and the scene classification pathways. For evaluating the robustness of our algorithm, we have assembled a challenging dataset consisting real-world images of eight different sport events, most of them collected from the Internet. Experimental results show that our hierarchical model performs better than existing methods.

### 6.1 Introduction and Motivation

One of the most remarkable feats of the human visual system is how rapidly, accurately and comprehensively it can recognize and understand the complex visual world. The various types of tasks related to understanding what we see in the visual world is called “visual recognition”. When presented with a real-world image, such as the top image of Fig 6.1, what do you see? It is a colorful image. On the top of

---

Li Fei-Fei  
Dept. of Computer Science, Stanford University, USA  
e-mail: [feifeili@cs.stanford.edu](mailto:feifeili@cs.stanford.edu)

Li-Jia Li  
Dept. of Computer Science, Stanford University, USA  
e-mail: [lijiali@cs.stanford.edu](mailto:lijiali@cs.stanford.edu)



**Fig. 6.1** Telling the *what, where and who* story. Given an *event* (rowing) image such as the one on the left, our system can automatically interpret what is the event, where does this happen and who (or what kind of objects) are in the image. The result is represented in the figure on the right. A red name tag over the image represents the event category. The scene category label is given in the white tag below the image. A set of name tags are attached to the estimated centers of the objects to indicate their categorical labels. As an example, from the right image, we can tell from the name tags that this is a rowing sport event held on a lake (scene). In this event, there are rowing boat, athletes, water and trees (objects).

the picture is mostly green color while the lower half is dominated by light blue, red and darker colors. There are salient edges in the foreground of the pictures, rows of round shapes paint a vivid visual picture in our mind. The different attributes of the images we describe, such as colors, edges, shapes, and textures, have been important research topics in the computer vision field. Recognizing these components of an image provide very important and useful information in a large number of practical applications. But this is not the level we communicate on and remember the visual world. It is also not the kind of description we would provide to a blind person. For most of us, this picture can be interpreted as a rich amount of semantically meaningful information. Now imagine the same scene, but this time I will describe it as a rowing event taking place on a lake. The water is clean and blue. Lush green trees stand along the shore of the lake in the background. A team of women athletes in red vests is training on a rowboat, accelerating to the right. I hope this time your mental imagery is much more vivid and meaningful than the first time. This is also the most natural way for most of us to interpret and describe our visual world. This kind of semantic interpretation of the visual world is called high-level visual recognition, part of the larger field known as vision. Vision is one of the most fundamental and important functionalities of an intelligence system. Humans rely on vision to survive, socialize and perform most of their daily tasks.

Recently, a psychophysics study has shown that in a single glance of an image, humans can not only recognize or categorize many of the individual objects in the scene, tell apart the different environments of the scene, but also perceive complex activities and social interactions [1]. In computer vision, a lot of progress has been made in object recognition and classification in recent years (see [2] for a review). A number of algorithms have also provided effective models for scene environment

categorization [3, 4, 5, 6]. But little has been done in event recognition in static images. In this work, we define an *event* to be a semantically meaningful human activity, taking place within a selected environment and containing a number of necessary objects. We present a first attempt to mimic the human ability of recognizing an event and its encompassing objects and scenes. Fig 6.1 best illustrates the goal of this work. We would like to achieve event categorization by as much semantic level image interpretation as possible. This is somewhat like what a school child does when learning to write a descriptive sentence of the event. It is taught that one should pay attention to the 5 W's: who, where, what, when and how. In our system, we try to answer 3 of the 5 W's: *what* (the **event** label), *where* (the **scene** environment label) and *who* (a list of the **object categories**).

Similar to object and scene recognition, event classification is both an intriguing scientific question as well as a highly useful engineering application. From the scientific point of view, much needs to be done to understand how such complex and high level visual information can be represented in efficient yet accurate way. In this work, we propose to decompose an event into its scene environment and the objects within the scene. We assume that the scene and the objects are independent of each other given an event. But both of their presences influence the probability of recognizing the event. We made a further simplification for classifying the objects in an event. Our algorithm ignores the positional and interactive relationships among the objects in an image. In other words, when athletes and mountains are observed, the event of rock climbing is inferred, in spite of whether the athlete is actually on the rock performing the climbing. Much needs to be done in both human visual experiments as well as computational models to verify the validity and effectiveness of such assumptions. From an engineering point of view, event classification is a useful task for a number of applications. It is part of the ongoing effort of providing effective tools to retrieve and search semantically meaningful visual data. Such algorithms are at the core of the large scale search engines and digital library organizational tools. Event classification is also particularly useful for automatic annotation of images, as well as descriptive interpretation of the visual world for visually-impaired patients.

## 6.2 Overall Approach

Our model integrates scene and object level image interpretation in order to achieve the final event classification. Let's use the sport game polo as an example. In the foreground, a picture of the polo game usually consists of distinctive objects such as horses and players (in polo uniforms). The setting of the polo field is normally a grassland. Following this intuition, we model an event as a combination of scene and a group of representative objects. The goal of our approach is not only to classify the images into different event categories, but also to give meaningful, semantic labels to the scene and object components of the images.

### 6.3 Literature Review

While our approach is an integrative one, our algorithm is built upon several established ideas in scene and object recognition. To the first order of approximation, an event category can be viewed as a scene category. Intuitively, a snowy mountain slope can predict well an event of skiing or snow-boarding. A number of previous works have offered ways of recognizing scene categories [4, 5, 6]. Most of these algorithms learn global statistics of the scene categories through either frequency distributions or local patch distributions. In the scene part of our model, we adopt a similar algorithm as Fei-Fei et al. [6]. In addition to the scene environment, event recognition relies heavily on foreground objects such as players and ball for a soccer game. Object categorization is one of the most widely researched areas recently. One could grossly divide the literature into those that use generative models (e.g., [7, 8, 9]) and those that use discriminative models or methods (e.g., [10, 11]). Given our goal is to perform event categorization by integrating scene and object recognition components, it is natural for us to use a generative approach. Our object model is adapted from the bag of words models that have recently shown much robustness in object categorization [12, 13, 14]. As [15] points out, other than scene and object level information, general layout of the image also contributes to our complex yet robust perception of a real-world image. Much can be included here for general layout information, from a rough sketch of the different regions of the image to a detailed 3D location and shape of each pixels of the image. We choose to demonstrate the usefulness of the layout/geometry information by using a simple estimation of 3 geometry cues: sky at infinity distance, vertical structure of the scene, and ground plane of the scene [16]. It is important to point out here that while each of these three different types of information is highly useful for event recognition (scene level, object level, layout level), our experiments show that we only achieve the most satisfying results by integrating all of them (Sec. 6.7).

Several previous works have taken on a more holistic approach in scene interpretation [17, 18, 19, 20]. In all these works, global scene level information is incorporated in the model for improving better object recognition or detection. Mathematically, our approach is closest in spirit with Sudderth et al [19]. We both learn a generative model to label the images. And at the object level, both of our models are based on the bag of words approach. Our model, however, differs fundamentally from the previous works by providing a set of integrative and hierarchical labels of an image, performing the *what*(event), *where*(scene) and *who*(object) recognition of an entire scene.

### 6.4 The Integrative Model

Given an image of an event, our algorithm aims to not only classify the type of event, but also to provide meaningful, semantic labels to the scene and object components of the images.

To incorporate all these different levels of information, we choose a generative model to represent our image. Fig 6.2 illustrates the graphical model representation. We first define the variables of the model, and then show how an image of a particular event category can be generated based on this model. For each image of an event, our fundamental building blocks are densely sampled local image patches (sampling grid size is  $10 \times 10$ ). In recent years, interest point detectors have demonstrated much success in object level recognition (e.g., [21][22][23]). But for a holistic scene interpretation task, we would like to assign semantic level labels to as many pixels as possible on the image. It has been observed that tasks such as scene classification benefit more from a dense uniform sampling of the image than using interest point detectors [5][6]. Each of these local image patches then goes on to serve both the scene recognition part of the model, as well as the object recognition part. For scene recognition, we denote each patch by  $X$  in Fig 6.2.  $X$  only encodes here appearance based information of the patch (e.g., a SIFT descriptor [21]). For the object recognition part, two types of information are obtained for each patch. We denote the appearance information by  $A$ , and the layout/geometry related information by  $G$ .  $A$  is similar to  $X$  in expression.  $G$  in theory, however, could be a very rich set of descriptions of the geometric or layout properties of the patch, such as 3D location in space, shape, and so on. For scenes subtending a reasonably large space (such as these event scenes), such geometric constraint should help recognition. In Sec 6.6, we discuss the usage of three simple geometry/layout cues: verticalness, sky at infinity and the ground-plane.<sup>1</sup>

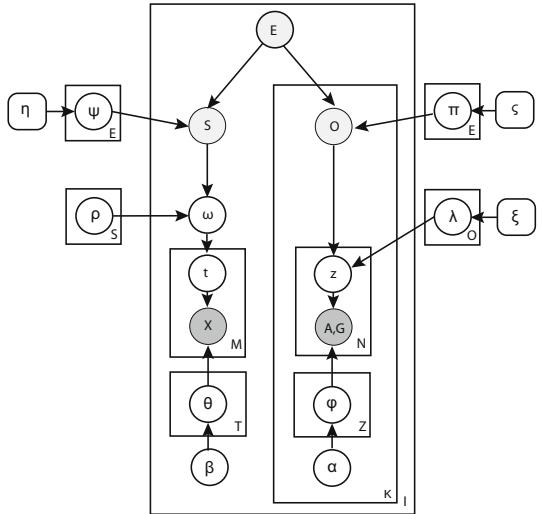
We now go over the graphical model (Fig 6.2) and show how we generate an event picture. Note that each node in Fig 6.2 represents a random variable of the graphical model. An open node is a latent (or unobserved) variable whereas a darkened node is observed during training. The lighter gray nodes (event, scene and object labels) are only observed during training whereas the darker gray nodes (image patches) are observed in both training and testing.

1. An **event** category is represented by the discrete random variable  $E$ . We assume a fixed uniform prior distribution of  $E$ , hence omitting showing the prior distribution in Fig 6.2. We select  $E \sim p(E)$ . The images are indexed from 1 to  $I$  and one  $E$  is generated for each of them.
2. Given the event class, we generate the **scene** image of this event. There are in theory  $S$  classes of scenes for the whole event dataset. For each event image, we assume only one scene class can be drawn.

---

<sup>1</sup> The theoretically minded machine learning readers might notice that the observed variables  $X$ ,  $A$  and  $G$  occupy the same physical space on the image. This might cause the problem of “double counting”. We recognize this potential confound. But in practice, since our estimations are all taken placed on the same “double counted” space in both learning and testing, we do not observe a problem. One could also argue that even though these features occupy the same physical locations, they come from different “image feature space”. Therefore this problem does not apply. It is, however, a curious theoretical point to explore further.

**Fig. 6.2** Graphical model of our approach. E, S, and O represent the event, scene and object labels respectively. X is the observed appearance patch for scene. A and G are the observed appearance and geometry/layout properties for the object patch. The rest of the nodes are parameters of the model. For details, please refer to Sec. 6.4.



- A scene category is first chosen according to  $S \sim p(S|E, \psi)$ .  $S$  is a discrete variable denoting the class label of the scene.  $\psi$  is the multinomial parameter that governs the distribution of  $S$  given  $E$ .  $\psi$  is a matrix of size  $E \times S$ , whereas  $\eta$  is an  $S$  dimensional vector acting as a Dirichlet prior for  $\psi$ .
  - Given  $S$ , we generate the mixing parameters  $\omega$  that governs the distribution of scene patch topics  $\omega \sim p(\omega|S, \rho)$ . Elements of  $\omega$  sum to 1 as it is the multinomial parameter of the latent topics  $t$ .  $\rho$  is the Dirichlet prior of  $\omega$ , a matrix of size  $S \times T$ , where  $T$  is the total number of the latent topics.
  - A patch in the scene image is denoted by  $X$ . To generate each of the  $M$  patches
    - Choose the latent topic  $t \sim \text{Mult}(\omega)$ .  $t$  is a discrete variable indicating which latent topic this patch will come from.
    - Choose patch  $X \sim p(X|t, \theta)$ , where  $\theta$  is a matrix of size  $T \times V_S$ .  $V_S$  is the total number of vocabularies in the scene codebook for  $X$ .  $\theta$  is the multinomial parameter for discrete variable  $X$ , whereas  $\beta$  is the Dirichlet prior for  $\theta$ .
3. Similar to the scene image, we also generate an **object** image. Unlike the scene, there could be more than one objects in an image. We use  $K$  to denote the number of objects in a given image. There is a total of  $O$  classes of objects for the whole dataset. The following generative process is repeated for each of the  $K$  objects in an image.
- An object category is first chosen according to  $O \sim p(O|E, \pi)$ .  $O$  is a discrete variable denoting the class label of the object. A multinomial parameter  $\pi$  governs the distribution of  $O$  given  $E$ .  $\pi$  is a matrix of size  $E \times O$ , whereas  $\varsigma$  is a  $O$  dimensional vector acting as a Dirichlet prior for  $\pi$ .
  - Given  $O$ , we are ready to generate each of the  $N$  patches  $A, G$  in the  $k^{\text{th}}$  object of the object image

- Choose the latent topic  $z \sim \text{Mult}(\lambda|O)$ .  $z$  is a discrete variable indicating which latent topic this patch will come from, whereas  $\lambda$  is the multinomial parameter for  $z$ , a matrix of size  $O \times Z$ .  $K$  is the total number of objects appear in one image, and  $Z$  is the total number of latent topics.  $\xi$  is the Dirichlet prior for  $\lambda$ .
- Choose patch  $A, G \sim p(A, G|t, \varphi)$ , where  $\varphi$  is a matrix of size  $Z \times V_O$ .  $V_O$  is the total number of vocabularies in the codebook for  $A, G$ .  $\varphi$  is the multinomial parameter for discrete variable  $A, G$ , whereas  $\alpha$  is the Dirichelet prior for  $\varphi$ . Note that we explicitly denote the patch variable as  $A, G$  to emphasize on the fact it includes both appearance and geometry/layout property information.

Putting everything together in the graphical model, we arrive at the following joint distribution for the image patches, the event, scene, object labels and the latent topics associated with these labels.

$$\begin{aligned} p(E, S, \mathbf{O}, \mathbf{X}, \mathbf{A}, \mathbf{G}, \mathbf{t}, \mathbf{z}, \omega | \rho, \varphi, \lambda, \psi, \pi, \theta) = \\ p(E) \cdot p(S|E, \psi) p(\omega|S, \rho) \prod_{m=1}^M p(X_m|t_m, \theta) p(t_m|w) \\ \cdot \prod_{k=1}^K p(O_k|E, \pi) \prod_{n=1}^N p(A_n, G_n|z_n, \varphi) p(z_n|\lambda, O_k) \end{aligned} \quad (6.1)$$

where  $\mathbf{O}, \mathbf{X}, \mathbf{A}, \mathbf{G}, \mathbf{t}, \mathbf{z}$  represent the generated objects, appearance representation of patches in the scene part, appearance and geometry properties of patches in the object part, topics in the scene part, and topics in the object part respectively. Each component of Eq. 6.1 can be broken into

$$p(S|E, \psi) = \text{Mult}(S|E, \psi) \quad (6.2)$$

$$p(\omega|S, \rho) = \text{Dir}(\omega|\rho_{\cdot}), S = j \quad (6.3)$$

$$p(t_m|\omega) = \text{Mult}(t_m|\omega) \quad (6.4)$$

$$p(X_m|t, \theta) = p(X_m|\theta_{\cdot}), t_m = j \quad (6.5)$$

$$p(O|E, \pi) = \text{Mult}(O|E, \pi) \quad (6.6)$$

$$p(z_n|\lambda, O) = \text{Mult}(z_n|\lambda, O) \quad (6.7)$$

$$p(A_n, G_n|z, \varphi) = p(A_n, G_n|\varphi_{\cdot}), z_n = j \quad (6.8)$$

where “.” in the equations represents components in the row of the corresponding matrix.

### 6.4.1 Labeling an Unknown Image

Given an unknown event image with unknown scene and object labels, our goal is: 1) to classify it as one of the event classes (*what*); 2) to recognize the scene environment class (*where*); and 3) to recognize the object classes in the image (*who*).

We realize this by calculating the maximum likelihood at the event level, the scene level and the object level of the graphical model (Fig 6.2).

At the object level, the likelihood of the image given the object class is

$$p(I|O) = \prod_{n=1}^N \sum_j P(A_n, G_n | z_j, O) P(z_j | O) \quad (6.9)$$

The most possible objects appear in the image are based on the maximum likelihood of the image given the object classes, which is  $O = \operatorname{argmax}_O p(I|O)$ . Each object is labeled by showing the most possible patches given the object, represented as  $O = \operatorname{argmax}_O p(A, G|O)$ .

At the scene level, the likelihood of the image given the scene class is:

$$p(I|S, \rho, \theta) = \int p(\omega|\rho, S) \left( \prod_{m=1}^M \sum_{t_m} p(t_m|\omega) \cdot p(X_m|t_m, \theta) \right) d\omega \quad (6.10)$$

Similarly, the decision of the scene class label can be made based on the maximum likelihood estimation of the image given the scene classes, which is  $S = \operatorname{argmax}_S p(I|S, \rho, \theta)$ . However, due to the coupling of  $\theta$  and  $\omega$ , the maximum likelihood estimation is not tractable computationally [24]. Here, we use the variational method based on Variational Message Passing [25] provided in [6] for an approximation.

Finally, the image likelihood for a given event class is estimated based on the object and scene level likelihoods:

$$p(I|E) \propto \sum_j P(I|O_j) P(O_j|E) P(I|S) P(S|E) \quad (6.11)$$

The most likely event label is then given according to  $E = \operatorname{argmax}_E p(I|E)$ .

## 6.5 Learning the Model

The goal of learning is to update the parameters  $\{\psi, \rho, \pi, \lambda, \theta, \beta\}$  in the hierarchical model (Fig 6.2). Given the event  $E$ , the scene and object images are assumed independent of each other. We can therefore learn the scene-related and object-related parameters separately.

We use Variational Message Passing method to update parameters  $\{\psi, \rho, \theta\}$ . Detailed explanation and update equations can be found in [6]. For the object branch of the model, we learn the parameters  $\{\pi, \lambda, \beta\}$  via Gibbs sampling [26] of the latent topics. In such a way, the topic sampling and model learning are conducted iteratively. In each round of the Gibbs sampling procedure, the object topic will be sampled based on  $p(z_i|\mathbf{z}_{\setminus i}, A, G, O)$ , where  $\mathbf{z}_{\setminus i}$  denotes all topic assignment except the current one. Given the Dirichlet hyperparameters  $\xi$  and  $\alpha$ , the distribution of topic given object  $p(z|O)$  and the distribution of appearance and geometry words

given topic  $p(A, G|z)$  can be derived by using the standard Dirichlet integral formulas:

$$p(z = i|\mathbf{z}_{\setminus i}, O = j) = \frac{c_{ij} + \xi}{\sum_i c_{ij} + \xi \times H} \quad (6.12)$$

$$p((A, G) = k|\mathbf{z}_{\setminus i}, z = i) = \frac{n_{ki} + \varphi}{\sum_k n_{ki} + \varphi \times V_O} \quad (6.13)$$

where  $c_{ij}$  is the total number of patches assigned to object  $j$  and object topic  $i$ , while  $n_{ki}$  is the number of patch  $k$  assigned to object topic  $i$ .  $H$  is the number of object topics, which is set to some known, constant value.  $V_O$  is the object codebook size. A patch is a combination of appearance ( $A$ ) and geometry ( $G$ ) features. By combining Eq 6.12 and 6.13, we can derive the posterior of topic assignment as

$$\begin{aligned} p(z_i|\mathbf{z}_{\setminus i}, A, G, O) &= p(z = i|\mathbf{z}_{\setminus i}, O) \times \\ &p((A, G) = k|\mathbf{z}_{\setminus i}, z = i) \end{aligned} \quad (6.14)$$

Current topic will be sampled from this distribution.

## 6.6 System Implementation

Our goal is to extract as much information as possible out of the event images, most of which are cluttered, filled with objects of variable sizes and multiple categories. At the feature level, we use a grid sampling technique similar to [6]. In our



**Fig. 6.3** Our dataset contains 8 sports event classes: rowing (250 images), badminton (200 images), polo (182 images), bocce (137 images), snowboarding (190 images), croquet (236 images), sailing (190 images), and rock climbing (194 images). In this figure, each triplet is randomly selected from our dataset. Our examples here demonstrate the complexity and diversity of this highly challenging dataset.

experiments, the grid size is  $10 \times 10$ . A patch of size  $12 \times 12$  is extracted from each of the grid centers. A 128-dim SIFT vector is used to represent each patch [21]. The poses of the objects from the same object class change significantly in these events. Thus, we use rotation invariant SIFT vector to better capture the visual similarity within each object class. A codebook is necessary in order to represent an image as a sequence of appearance words. We build a codebook of 300 visual words by applying K-means for the 200000 SIFT vectors extracted from 30 randomly chosen training images per event class. To represent the geometry/layout information, each pixel in an image is given a geometry label using the codes provided by [18]. In this approach, only three simple geometry/layout properties are used. They are: ground plane, vertical structure and sky at infinity. Each patch is assigned a geometry membership by the major vote of the pixels within.

## 6.7 Experiments and Results

### 6.7.1 Dataset

As the first attempt to tackle the problem of static event recognition, we have no existing dataset to use and compare with. Instead we have compiled a new dataset containing 8 sports event categories collected from the Internet: bocce, croquet, polo, rowing, snowboarding, badminton, sailing, and rock climbing. The number of images in each category varies from 137 (bocce) to 250 (rowing). As shown in Fig. 6.3, this event dataset is a very challenging one. Here we highlight some of the difficulties.

- The background of each image is highly cluttered and diverse;
- Object classes are diverse;
- Within the same category, sizes of instances from the same object are very different;
- The pose of the objects can be very different in each image;
- Number of instances of the same object category change diversely even within the same event category;
- Some of the foreground objects are too small to be detected.

We have also obtained a thorough groundtruth annotation for every image in the dataset (in collaboration with Lotus Hill Research Institute [27]). This annotation provides information for: event class, background scene class(es), most discernable object classes, and detailed segmentation of each objects.

### 6.7.2 Experimental Setup

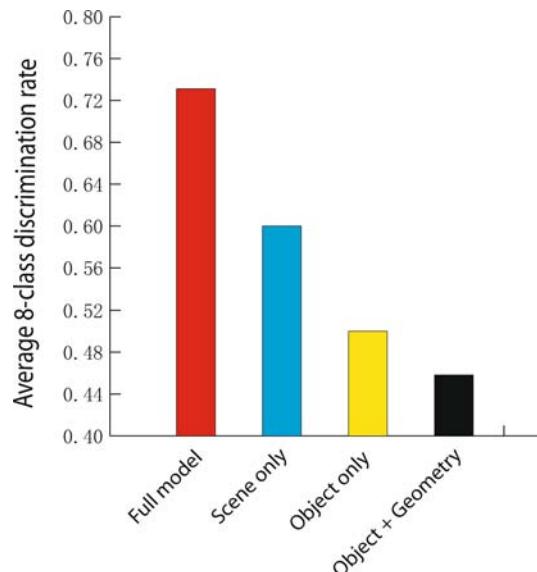
We set out to learn to classify these 8 events as well as labeling the semantic contents (scene and objects) of these images. For each event class, 70 randomly selected images are used for training and 60 are used for testing. We do not have any previous

**Fig. 6.4** Confusion table for the 8-class event recognition experiment. The average performance is 73.4%. Random chance would be 12.5%.

		Average Perf.= 73.4%						
		bocce	.02	.17	.05	.25		
bocce		.52	.92				.03	.05
badminton		.27		.62	.02		.10	
polo		.03	.02	.03	.80		.12	
rowing					.77		.03	.02
snowboarding			.18					
croquet		.27	.03	.07	.12	.52		
sailing			.13			.07	.80	
rockclimbing		.05			.02		.02	.92
bocce	bocce							
badminton	badminton							
polo	polo							
rowing	rowing							
snowboarding	snowboarding							
croquet	croquet							
sailing	sailing							
rockclimbing	rockclimbing							

work to compare to. But we test our algorithm and the effectiveness of each components of the model. Specifically, we compare the performance of our full integrative model with the following baselines.

- A *scene only* model. We use the LDA model of [6] to do event classification based on scene categorization only. We “turn off” the influence of the object part by setting the likelihood of  $O$  in Eq 6.11 to a uniform distribution. This is effectively a standard “bag of words” model for event classification.
- An *object only* model. In this model we learn and recognize an event class based on the distribution of foreground objects estimated in Eq 6.9. No geometry/layout



**Fig. 6.5** Performance comparison between the full model and the three control models (defined in Sec. 6.7.2). The x-axis denotes the name of the model used in each experiment. The ‘full model’ is our proposed integrative model (see Fig 6.2). The y-axis represents the average 8-class discrimination rate, which is the average score of the diagonal entries of the confusion table of each model.



**Fig. 6.6** (This figure is best viewed in color and with PDF magnification.) Image interpretation via event, scene, and object recognition. Each row shows results of an event class. **Column 1** shows the event class label. **Column 2** shows the object classes recognized by the system. Masks with different colors indicate different object classes. The name of each object class appears at the estimated centroid of the object. **Column 3** is the scene class label assigned to this image by our system. Finally **Column 4** shows the sorted object distribution given the event. Names on the x-axis represents the object class, the order of which varies across the categories. y-axis represents the distribution.

information is included. We “turn off” the influence of the scene part by setting the likelihood of  $S$  in Eq 6.11 to a uniform distribution.

- A *object + geometry* model. Similar to the object-only model, here we include the feature representations of both appearance ( $A$ ) and geometry/layout ( $G$ ).

Except for the LDA model, training is supervised by having the object identities labeled. We use exactly the same training and testing images in all of these different model conditions.

### 6.7.3 Results

We report an overall 8-class event discrimination of 73.4% by using the full integrative model. Fig 6.4 shows the confusion table results of this experiment. In the confusion table, the rows represent the models for each event category while the columns represent the ground truth categories of events. It is interesting to observe that the system tends to confuse bocce and croquet, where the images tend to share similar foreground objects. On the other hand, polo is also more easily confused with bocce and croquet because all of these events often take places in grassland type of environments. These two facts agree with our intuition that an event image could be represented as a combination of the foreground objects and the scene environment.

In the control experiment with different model conditions, our integrative model consistently outperforms the other three models (see Fig 6.5). A curious observation is that the *object + geometry* model performs worse than the *object only* model. We believe that this is largely due to the simplicity of the geometry/layout properties. While these properties help to differentiate sky, ground from vertical structures, they also introduce noise. As an example, water and snow are always incorrectly classified as sky or ground by the geometry labeling process, which deteriorates the result of object classification. However, the scene recognition alleviates the confusion among water, snow, sky and ground by encoding explicitly their different appearance properties. Thus, when the scene pathway is added to the integrated model, the overall results become much better.

Finally, we present more details of our image interpretation results in Fig 6.6. At the beginning of this chapter, we set out to build an algorithm that can tell a *what*, *where* and *who* story of the sport event pictures. We show here how each of these W’s is answered by our algorithm. Note all the labels provided in this figure are automatically generated by the algorithm, no human annotations are involved.

## 6.8 Conclusion

Semantic interpretation of the visual world is an indispensable functionality of the future generations of artificial intelligence system. This project aims to contribute to both the scientific questions of image modeling and the technological advancement of visual intelligence. One of the most important applications is personal

assistance to visually-impaired or blind people. Currently, other than specific domain applications such as texts and faces, little technology is available to assist them to interpret and analyze the visual environment in a comprehensive and meaningful way. Semantic understanding of images could serve to advance the state of the art assistance in this domain. It will also improve real-word applications that require advanced visual recognition tools. One example is the increasing need for sophisticated and meaningful sorting and searching tools for large image datasets, such as personal photo collections and images on the internet. Our model is, of course, just the first attempt for such an ambitious goal.

## References

1. Fei-Fei, L., Iyer, A., Koch, C., Perona, P.: What do we perceive in a glance of a real-world scene? *Journal of Vision* 7(1), 10, 1–29 (2007)
2. Fei-Fei, L., Fergus, R., Torralba, A.: Recognizing and learning object categories. In: Short Course of the International Conference on Computer Vision and Pattern Recognition (2007),  
<http://people.csail.mit.edu/torralba/shortCourseRLOC/index.html>
3. Szummer, M., Picard, R.: Indoor-outdoor image classification. In: Proceedings of International Workshop on Content-based Access of Image and Video Databases (1998)
4. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision* 42 (2001)
5. Vogel, J., Schiele, B.: A semantic typicality measure for natural scene categorization. In: Rasmussen, C.E., Bülthoff, H.H., Schölkopf, B., Giese, M.A. (eds.) DAGM 2004. LNCS, vol. 3175, pp. 195–203. Springer, Heidelberg (2004)
6. Fei-Fei, L., Perona, P.: A Bayesian hierarchy model for learning natural scene categories. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2005)
7. Weber, M., Welling, M., Perona, P.: Unsupervised learning of models for recognition. In: Vernon, D. (ed.) ECCV 2000. LNCS, vol. 1842, pp. 101–108. Springer, Heidelberg (2000)
8. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 264–271 (2003)
9. Kumar, M.P., Torr, P.H.S., Zisserman, A.: Obj cut. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 18–25 (2005)
10. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 511–518 (2001)
11. Zhang, H., Berg, A., Maire, M., Malik, J.: Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2006)
12. Csurka, G., Bray, C., Dance, C., Fan, L.: Visual categorization with bags of keypoints. In: International Workshop on Statistical Learning in Computer Vision, ECCV, pp. 1–22 (2004)

13. Sivic, J., Russell, B., Efros, A., Zisserman, A., Freeman, W.: Discovering object categories in image collections. In: Proceedings of the International Conference on Computer Vision (2005)
14. Li, L.-J., Wang, G., Fei-Fei, L.: Optimol: automatic online picture collection via incremental model learning. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
15. Wolfe, J.: Visual memory: what do you know about what you saw? *Current Biology* 8, R303–R304 (1998)
16. Hoiem, D., Efros, A., Hebert, M.: Automatic photo pop-up. In: Proceedings of ACM SIGGRAPH, vol. 24(3), pp. 577–584 (2005)
17. Murphy, K., Torralba, A., Freeman, W.: Using the forest to see the trees:a graphical model relating features, objects and scenes. In: Proceedings of Neural Information Processing Systems (2004)
18. Hoiem, D., Efros, A., Hebert, M.: Putting Objects in Perspective. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2006)
19. Sudderth, E., Torralba, A., Freeman, W., Willsky, A.: Learning hierarchical models of scenes, objects, and parts. In: Proceedings of the International Conference on Computer Vision (2005)
20. Tu, Z., Chen, X., Yuille, A., Zhu, S.: Image Parsing: Unifying Segmentation, Detection, and Recognition. *International Journal of Computer Vision* 63(2), 113–140 (2005)
21. Lowe, D.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision (1999)
22. Dorko, G., Schmid, C.: Object class recognition using discriminative local features. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (submitted)
23. Obdrzalek, S., Matas, J.: Object recognition using local affine frames on distinguished regions. In: Proceedings of the British Machine Vision Conference, pp. 113–122 (2002)
24. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet allocation. *Journal of Machine Learning Research* 3, 993–1022 (2003)
25. Winn, J., Bishop, C.M.: Variational message passing. *Journal of Machine Learning Research* 6, 661–694 (2004)
26. Krempp, S., Geman, D., Amit, Y.: Sequential learning with reusable parts for object detection. Technical report, Johns Hopkins University (2002)
27. Yao, Z.-Y., Yang, X., Zhu, S.-C.: Introduction to a large scale general purpose groundtruth dataset: methodology, annotation tool, and benchmarks. In: Yuille, A.L., Zhu, S.-C., Cremers, D., Wang, Y. (eds.) *EMMCVPR 2007. LNCS*, vol. 4679, pp. 169–183. Springer, Heidelberg (2007)

# Chapter 7

## Semantic Texton Forests

Matthew Johnson and Jamie Shotton

**Abstract.** The *semantic texton forest* is an efficient and powerful low-level feature which can be effectively employed in the semantic segmentation of images. As ensembles of decision trees that act directly on image pixels, semantic texton forests do not need the expensive computation of filter-bank responses or local descriptors. They are extremely fast to both train and test, especially compared with  $k$ -means clustering and nearest-neighbor assignment of feature descriptors. The nodes in the trees provide (i) an implicit hierarchical clustering into semantic textons, and (ii) an explicit local classification estimate. The *bag of semantic textons* combines a histogram of semantic textons over an image region with a region prior category distribution. The bag of semantic textons can be used by an SVM classifier to infer an image-level prior over categories, allowing the segmentation to emphasize those categories that the SVM believes to be present. We will examine the segmentation performance of semantic texton forests on two datasets including the VOC 2007 segmentation challenge.

### 7.1 Introduction

In this chapter we examine *semantic texton forests*, and evaluate their use for image categorization and semantic segmentation. Semantic texton forests (STFs) demonstrate that one can build powerful texton codebooks without computing expensive filter-banks or descriptors, and without performing costly  $k$ -means clustering and nearest-neighbor assignment. They are randomized decision forests that use only simple pixel comparisons on local image patches, performing both an implicit

---

Matthew Johnson  
Nokia, San Francisco, USA  
e-mail: matthew.3.johnson@nokia.com

Jamie Shotton  
Microsoft Research, Cambridge, UK  
e-mail: jamiesho@microsoft.com

hierarchical clustering into semantic textons and an explicit local classification of the patch category. STFs provide advantages over other algorithms in both quantitative performance and execution speed.

We will look at two applications of STFs: image categorization (inferring the object categories present in an image) and semantic segmentation (dividing the image into coherent regions and simultaneously categorizing each region). The tool that will be used for both of these is the *bag of semantic textons*. This is computed over a given image region, and extends the bag of words model [6] by combining a hierarchical histogram of the semantic textons with a prior category distribution. By considering the image as a whole, a highly discriminative descriptor for categorization can be obtained. For segmentation, a bag of semantic textons can be computed for many local rectangular regions and then a second randomized decision forest can be built which achieves efficient and accurate segmentation by drawing on appearance and semantic context.

The segmentation algorithm depends on image information that even with semi-local context can often be ambiguous. The global statistics of the image, however, can be more discriminative and may be sufficient to accurately estimate the image categorization. It is therefore useful to use categorization as an *image-level prior* to improve segmentation by emphasizing the categories most likely to be present.

## 7.2 Related Work

Textons [17, 35] and visual words [32] have proven powerful discrete image representations for categorization and segmentation [6, 30, 39, 40]. We will treat the terms texton and visual word synonymously. Filter-bank responses (derivatives of Gaussians, wavelets, etc.) or invariant descriptors (e.g., SIFT [16]) are computed across a training set, either at sparse interest points [19] or more densely; recent results in [22] suggest that densely sampling visual words improves categorization performance. The collection of descriptors are then clustered to produce a codebook of visual words, typically with the simple but effective  $k$ -means, followed by nearest-neighbor assignment. Unfortunately, this three-stage process is extremely slow and often the most time consuming part of the whole algorithm, even with optimizations such as k-d trees, the triangle inequality [7], or hierarchical clusters [21, 29].

The recent work of Moosmann *et al* [20] proposed a more efficient alternative, in which training examples are recursively divided using a randomized decision forest [1, 10] and where the splits in the decision trees are comparisons of a descriptor dimension to a threshold. Semantic texton forests extend [20] in three ways: (i) the model is learned *directly* from the image pixels, bypassing the expensive step of computing image descriptors; (ii) while [20] use the learned decision forest only for clustering, here it is used as a classifier, which enables the algorithm to use semantic context for image segmentation; and (iii) in addition to the leaf nodes used in [20], split nodes as hierarchical clusters are included. A related method, the pyramid match kernel [11], exploits a hierarchy in descriptor space, though it

requires the computation of feature descriptors and is only applicable to kernel based classifiers.

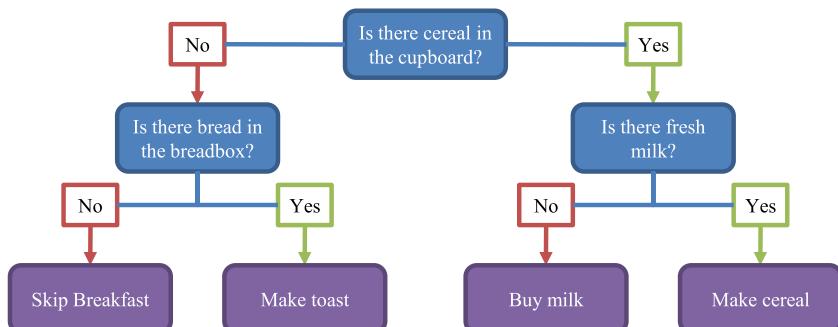
The pixel-based features used are similar to those in [15], but the forests are trained to recognize object categories, not match particular feature points.

Other work has also looked at alternatives to  $k$ -means. Recent work [34] quantizes feature space into a hyper-grid, but requires descriptor computation and can result in very large visual word codebooks. Winder & Brown [38] learned the parameters of generic image descriptors for 3D matching, though did not address visual word clustering. Jurie & Triggs [13] proposed building codebooks using mean shift, but did not incorporate semantic information in the codebook generation.

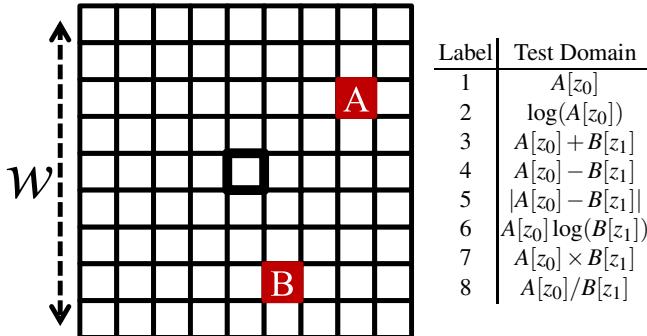
### 7.3 Randomized Decision Forests

The randomized decision forest is a fast and accurate classifier [1, 10] that is an ensemble of decision trees. Decision trees are a construct used extensively in data mining [5] and machine learning [4], and consist of a hierarchy of questions, as illustrated in Figure 7.1. A tree is traversed, starting at the root, by repeatedly asking questions and branching to the relevant child node until a leaf node is reached. At the leaf, the stored decision is output. In this chapter, decision trees are used to categorize individual image pixels, and so the questions, or “split tests”, at each node are based on image information. The specific tests used in this work are shown in Figure 7.2.

A randomized decision forest combines the output of many different decision trees, each of which has a different structure and split tests. The term “randomized” refers to the training algorithm in two ways. Firstly, each tree is trained on a random subset of the data following the method outlined in Section 7.3.1.1. Secondly, when building the tree, several candidate split tests are chosen at random from a large



**Fig. 7.1** Example Decision Tree. This is an example decision tree for determining what to do about breakfast. At each node a simple split test is performed, and the result of that test is used to determine which child to choose. This process is repeated until a leaf node is reached, with each leaf encoding a particular decision to be made that is based upon all of the tests performed to reach that node.



**Fig. 7.2** Pixel Comparison Split Tests. The split tests in a semantic texton forest consist of pixel combinations within a square neighborhood centered on the pixel to be categorized of size  $w \times w$ .  $z_0$  and  $z_1$  are channels in the image, e.g., R, G and B in RGB images. It is not necessary that  $z_0 = z_1$ .

pool of potential features, and the test that optimally splits the data (under some optimization criterion) is taken. These two forms of randomization help generalization by ensuring that no two trees in the forest can overfit to the whole training set.

Let us formalize notation. For the forests described in this chapter, the goal is to determine the category  $c$  of a pixel  $p$ , given the context around that pixel. We assume a labeled training set, such as that in Figure 7.4. Each forest contains trees with nodes  $n$ , and leaf nodes  $l$ . Associated with each node is a learned category distribution  $P(c|n)$ . An example semantic texton tree can be seen in Figure 7.3, in which a tree has been trained on sheep and grass images and can effectively segment an image according to these two semantic categories.

When a new pixel is to be classified, the whole forest achieves an accurate and robust classification by averaging the class distributions over the leaf nodes  $L(p) = (l_1, \dots, l_T)$  reached by the pixel  $p$  for all  $T$  trees:

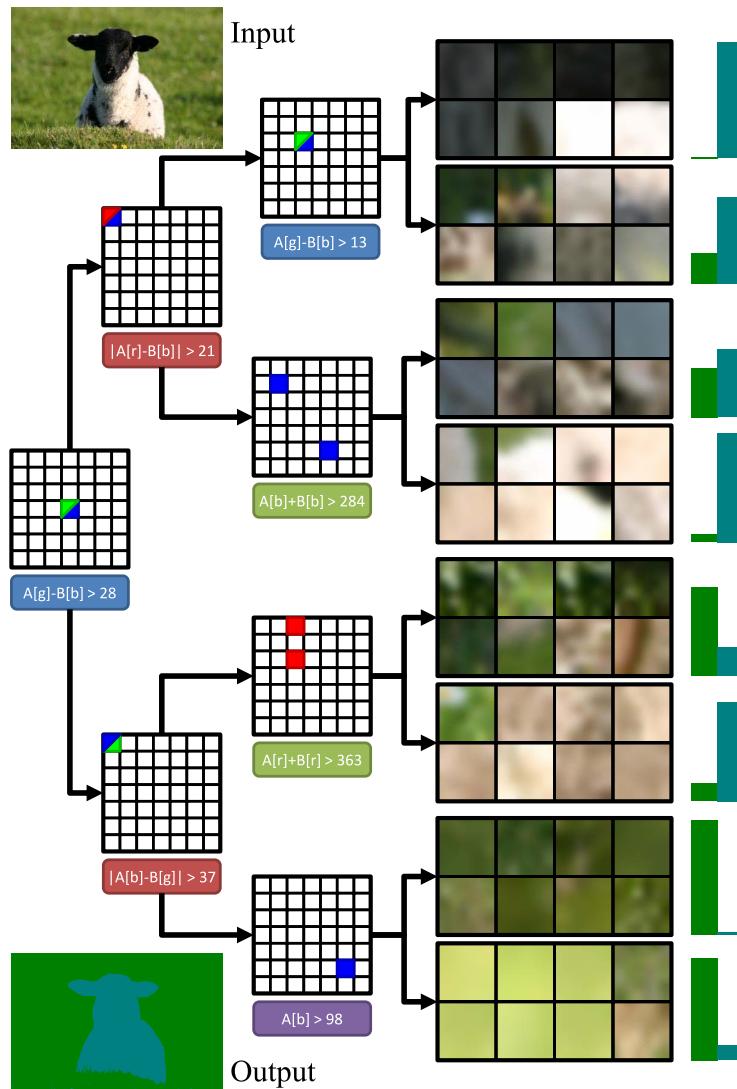
$$P(c|L(p)) = \sum_{t=1}^T P(c|l_t)P(t). \quad (7.1)$$

An example of the overall structure of the forest can be seen in Figure 7.5.

Existing work has shown the power of decision forests as either classifiers [3, 15, 18] or a fast means of clustering descriptors [20]. In this chapter we will examine an extended model in which the forest is used both for classification and for a hierarchical clustering.

### 7.3.1 Training the Forest

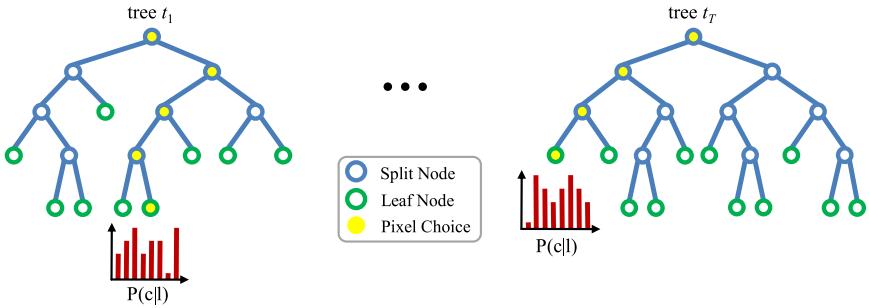
Each tree in the forest is build separately on a subset of the training images. We describe below how each tree is built greedily, and the parameter settings that will



**Fig. 7.3** Sample Semantic Texton Tree. This is an actual semantic texton tree, trained on 23 images of grass and sheep as described in Section 7.3.1.1. The split tests are shown at each split node as the image patch ( $w = 7$ ). The two triangles indicate the offset pixel location (relative to the center of the grid) and their colors indicate the image color channel used for this split test. The leaf nodes are represented by 8 patches sampled from the training pixels which reached those nodes and the distribution  $P(x|c)$  for that leaf node where green represents grass and blue represents sheep. The input image is an unseen test image, with the resulting semantic segmentation shown below.



**Fig. 7.4** Training Data. A tree is trained on ground-truth labeled images like these above from the MSRC dataset [30], in which each pixel is labelled with an object category (indicated by colors).



**Fig. 7.5** Semantic Texton Forest Structure. The forest is made up of  $T$  binary trees. Each split node  $n$  in a tree (blue circles) has a test associated with it, and based upon the result of that test one or the other child is chosen. When a leaf node  $l$  in a tree  $t$  is reached (green circles), the  $P(c|l_t)$  distribution for that leaf is used as a soft category decision for the test pixel. In this figure, a sample decision path for each tree is denoted by a series of yellow circles. The final decision is a combination of  $P(c|l_t)$  for all  $t \in T$ .

affect this. We also discuss how to learn invariances to rotation, scale and other fundamental image transformations.

### 7.3.1.1 Building a Tree

The training data consists of a set  $P$  of pixels sampled from training images at every 4th pixel, and ignoring pixels marked as background. This sub-sampling decreases the time required for tree construction. To ensure good estimates of the tree class distributions, *all* pixels are used later to “fill” the tree after construction as described below in Section 7.3.1.3.

Each tree is constructed by recursively partitioning  $P$  into two subsets  $P_{left}$  and  $P_{right}$  based upon a split test.  $P_{left}$  is used to create the left subtree and  $P_{right}$  is used for the right, repeating the process until a stopping condition is met. The split test used to partition  $P$  is chosen in the same manner as [15]: by searching over a set of

possible tests and choosing that which maximizes the expected gain in information about the node categories. The information gain is calculated as

$$\Delta E = -\frac{|P_{left}|}{|P|}E(P_{left}) - \frac{|P_{right}|}{|P|}E(P_{right}), \quad (7.2)$$

where  $E(I)$  is the Shannon entropy of the classes in the set of examples  $P$ .

### 7.3.1.2 Parameters

Training a randomized decision forest involves several parameters, including:

*Type of Split Tests.* The types of split tests used can play a significant role in tree training and performance, with different tests acting in a complimentary manner.

*Number of Trees.* The number of trees in the forest is a tradeoff between accuracy and speed.

*Information Channels.* The number and type of information channels in the image and how they are processed can have a large impact on which tests should be chosen and on model generalization. In the case of color images, this takes the form of the encoding method chosen for the color at each pixel.

*Maximum Tree Depth.* Deeper trees are more powerful classifiers, but more prone to overfitting.

*Value of Window Size.* The size of the window around each pixel  $w$  can effect whether the tree learns local image characteristics or contextual information. Larger windows provide more discriminative features but ones that are less likely to generalize.

The choice of these parameters depends heavily on the nature of the dataset, and should be optimized against a validation set. Smaller datasets will tend to need many shallower trees, whereas larger datasets make generalization easier and so fewer deeper trees may work well. We discuss below the results of several experiments designed to discover the best parameters for the task of pixel-level category inference. The cost of performing a full exploration of the parameter space is prohibitive, and so our exploration varies one parameter while holding the others constant.

Of particular interest are the types of split tests made available to the training algorithm. The pixel tests listed in Figure 7.2 are those used in the experiments below, where  $A[z_0]$  and  $B[z_1]$  are the values of pixels within a patch of size  $w \times w$  centered on the training pixel. The channels  $z_0$  and  $z_1$  do not have to be the same. While some test types have a basis in image structure (the difference and absolute difference of pixels is invariant to a global intensity shift, and signifies edges in the image), others are just possible discriminative combinations of pixels. In addition to these pixel tests, we evaluate two rectangle-based tests: the Haar-like features of [37] and the rectangle sum features of [30].

### 7.3.1.3 Supervision

Labeled image data is used to train a semantic texton forest, consisting pairs  $(p, c)$  of pixels and category labels. In the case of *full supervision* each pixel is given a

training label as shown in Figure 7.4. During training, the distribution  $P(c|n)$  is computed as a normalized histogram of the training tuples which reached a particular node  $n$ :

$$P(c|n) = \frac{H_n[c]}{\sum_c H_n[c]}, \quad (7.3)$$

where  $H_n[c]$  is the number of pixels of class  $c$  that passed through a node  $n$  during training. The process of computing this histogram at each node will be referred to as “filling” the forest. Filling is performed using all of the pixels in the training data, by passing each pixel down a tree and incrementing the relevant histogram bin  $H_n[c]$ . If desired, a small Dirichlet prior corresponding to a extra constant count added to all classes can be used to smooth the distributions.

In the case of *partial supervision*, we do not have pixel labels, but rather the set of categories present somewhere in the image. In other words, we have just the distribution  $P(c|x)$  where  $x$  is an observed *topic* for the image. In this circumstance, the topic can be thought of as an underlying meaning generating the categories in the image, for example a “forest” topic is more likely to produce “tree” pixels, whereas a “city” topic would be more likely to produce “building” pixels. As we have no data about  $P(p|c)$ , it is modeled as a uniform distribution. Thus, to create training points to use in a partially supervised forest we first sample a category using  $P(c|x)$  and then sample a pixel using  $P(p|c)$ . The forest is subsequently trained on these points, and the result has a fairly low pixel accuracy (though still greater than random chance).

### 7.3.1.4 Learning Invariances

Although using raw pixels as features is much faster than first computing descriptors or filter-bank responses, one risks losing their inherent invariances. Thus, as the distribution  $P(c|n)$  is estimated with the training images these images are augmented with copies that are artificially transformed geometrically and photometrically as done by Lepetit in [15]. This allows the forest to *learn* the right degree of invariance required for a particular problem. In these experiments the transformations used were rotation, scaling, and left-right flipping as geometric transformations, as well as affine photometric transformations.

### 7.3.2 Experiments

In the following experiments, accuracy was measured as the mean percentage of pixels labeled correctly over all categories. A confusion matrix  $M$  was computed for each image over pixels  $p \in P_R$ , where  $P_R$  is the set of test pixels. Thus, the individual cells of  $M$  are computed as

$$M[i, j] = |\{p : p \in P_R, G(p) = c_i, \text{argmax}_c P(c|L_p) = c_j\}|, \quad (7.4)$$

using the image ground-truth  $G$ . For these experiments the mean category accuracy  $\mu$  is reported, calculated as

$$\mu = \frac{1}{Z} \sum_{i=1}^Z \alpha_i \quad (7.5)$$

where  $Z$  is the number of categories and

$$\alpha_i = \frac{M[i, i]}{\sum_j M[i, j]} . \quad (7.6)$$

A second metric is the overall accuracy  $\alpha$ , calculated as

$$\alpha = \frac{\sum_i M[i, i]}{\sum_i \sum_j M[i, j]} . \quad (7.7)$$

The mean category accuracy  $\mu$  ensures a fair balance across categories which potentially have very different numbers of pixels in the data. The overall accuracy  $\alpha$ , on the other hand, tells us what proportion of the image the system can reliably segment. Both are important to get a sense of accuracy of the system, e.g., a high  $\alpha$  and low  $\mu$  indicates overfitting to a particular category which is disproportionately represented in the dataset.

The control training scenario was set as the following parameters:

Parameter	Value
<i>Number of Trees</i>	5
<i>Maximum Depth</i>	10
<i>Type of Split Tests</i>	$A, A + B, A \log(B),  A - B $
$w$	10
<i>Color Channels</i>	CIELab
<i>Data % Per Tree</i>	25

In each experiment, a single training parameter was changed while keeping all others constant to see the effect it had on test performance. Experiments were performed on the MSRC21 dataset [30]. In each experiment, ten trees were trained on a subset of the data and then filled with all of the data points as described in Section 7.3.1.4. Ten forests of five trees (with the exception of the number of trees experiment) were then created from permutations of these ten trees. The reported values are the mean  $\alpha$  and  $\mu$  of 10 segmentation trials run with those forests over the test data after being trained on the training and validation data with the specified parameters. Error bars indicating the standard error are omitted due to the remarkably low error on the values making them indiscernible. It is quite possibly due to the fact that the different forests being used for each trial are 5 trees chosen from the same set of 10, but even so it is intriguing to note that 10 different trees trained independently on different subsets of the data but with the same parameters achieved very close to the same accuracy on the test data.

In Figure 7.6, one can see the effect different combinations of feature tests has on segmentation performance. In each of the five trials a different random order of five feature tests was chosen, shown below in Table 7.2, and in each step an additional feature was added for the tree to use, accompanied by an increase in the feature pool

**Table 7.1** Test Proportions for MSRC21 Dataset. A semantic texton forest was trained on the MSRC21 dataset, making all of the tests in the table available to it during training. The counts for each test were recorded over the entire forest to get a sense of which tests were most useful for categorization. As can be seen, the rectangle-based features performed best and were chosen at a much higher rate than the others. Note that the Haar features are a superset of the additive and subtractive pixel features.

	Counts	%
Rectangle	670930	39.89%
Haar	316368	18.81%
$A$	179816	10.69%
$A + B$	118638	7.05%
$A \times B$	107146	6.37%
$ A - B $	105794	6.29%
$A \log(B)$	92274	5.49%
$A - B$	45968	2.73%
$A / B$	44954	2.67%
<i>Total</i>	1681888	

size. One trial was done with every test and a pool size of 1000 (the rightmost bar on the graph) showing the practical limit on accuracy. Every additional test made available to the algorithm will usually improve performance, but there are certain groups of tests which work together better than others. So it is that performance can actually drop, sometimes dramatically, when a new test is added that elsewhere when added improved performance. The ideal mixture of tests depends to a certain extent on the data, and these experiments should only be considered in so far as they show the rate at which adding different feature tests improves performance. One good method of finding out which tests work best for a dataset is to train one forest with a large pool size and every available test. As each tree will choose tests based purely on information gain, this will give a very good indicator of which tests work best for the dataset, and a subset can be chosen either for quicker training or to optimize tree performance depending on the situation at hand. As an example, Table 7.1 gives the proportions of tests for the forest which had every test available to it, showing that for the evaluation dataset rectangle features like the rectangle sum [30] and Haar-like features [37] perform very well.

In Figure 7.7 the effect of the number of trees in the forest can be seen. It is clear that there are diminishing returns as forest size increases. In order to investigate the effects of different methods of representing color on performance forests were trained on grayscale, RGB, CIELab, and HSV images, the results of which can be seen in Figure 7.8. CIELab clearly results in the best performance, most likely due to its useful features (e.g., meaningful perceptual distances, device independence) for computer vision as described in [12]. The effect of different tree depths is shown in Figure 7.9. The values chosen for this trial were dependent on the dataset size, as the amount of data needed to fill a tree increases exponentially with tree depth, but it can be seen that as the number of nodes in the tree increases so does its classification ability, which is to be expected due to the way in which the trees are trained

**Table 7.2** Test Domain Experimental Setup. The size of the feature pool and the number of different feature domains those features could be drawn from increases from right to left, with 25 total trials being performed. Results are shown in Figure 7.6.

Trial	1	2	3	4	5
1	$A$	$A \times B$	$A - B$	Haar	$ A - B $
2	$ A - B $	$A - B$	$A/B$	$A$	$\log(A)$
3	$A/B$	$A - B$	$\log(A)$	$A \times B$	$A$
4	$\log(A)$	Rectangle	$A - B$	$ A - B $	$A + B$
5	$A \times B$	$A$	$A + B$	$A \log(B)$	$A/B$
Pool Size	100	200	300	400	500

(i.e. nodes will stop splitting if there is no expected information gain). Finally, the effect of the  $w$  parameter (the size of the window test pixels can be chosen from) can be seen in Figure 7.10. The effect here of a steady increase in average accuracy coupled with an increase and then decline in overall accuracy is very interesting. This is likely due to the fact that the easier categories which take up many of the pixels in the dataset (e.g., grass and sky) do not require the context of nearby pixels to be classified, but the smaller and more complex categories can take advantage of nearby information to aid in classification that is only possible when the window of possible pixels is significantly increased.

## 7.4 Image Categorization

The bag of words histogram has been extensively used in recent years for object categorization in computer vision [32, 6, 9, 25, 31, 28, 40]. As an alternative to the typical method for creating these histograms (using interest points, descriptors and a vector quantized patch dictionary) one can use the localized bag of semantic textons (BOST), illustrated in Figure 7.11. This extends the bag of words representation with low-level semantic information, as follows.

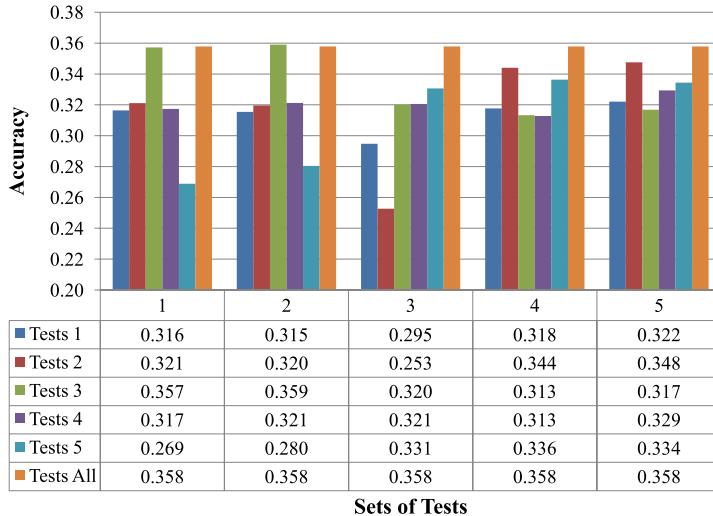
Given for each pixel  $p$  the leaf nodes  $L(p) = (l_1, \dots, l_T)$  and inferred class distribution  $P(c|L(p))$ , one can compute over image region  $r$ :

1. A non-normalized histogram  $H_r(n)$  that concatenates the occurrences of tree nodes  $n$  across the different trees [20], and
2. A conditional distribution over the region given by the average class distribution  $P(c|r) = \sum_{p \in r} P(c|L_p)P(p)$ .

Experiments were performed with tree histograms (unlike the leaf histograms of [20]) where both leaf nodes  $l$  and split nodes  $n$  are included in the histogram, such that

$$H_r(n) = \sum_{n' \in \text{child}(n)} H_r(n') . \quad (7.8)$$

This histogram therefore uses the hierarchy of clusters implicit in each tree. Each  $P(c|L(p))$  is already averaged across trees, and hence there is a single region prior  $P(c|r)$  for the whole forest.



**Fig. 7.6** Effect of Different Test Domains. The values shown are the mean over ten trials of the mean category accuracy ( $\mu$ ). The computation for  $\mu$  is described in Section 7.3.2. This graph tracks the effect of increasing the test domains on accuracy, with the overall trend being that the larger the domain of tests the more accurate the system becomes, though there is quite a lot of variation with some compositional changes, particularly in set 3. For reference, a system trained with all possible tests and a pool size of 1000 is shown as the rightmost bar. The meanings of the numbers in the graph are explained in Table 7.2.

#### 7.4.1 Tree Histograms and Pyramid Matching

Consider first the BOST histogram computed for just one tree in the STF. The kernel function (based on  $\|\cdot\|$ ) is then

$$K(P, Q) = \frac{1}{\sqrt{Z}} \tilde{K}(P, Q), \quad (7.9)$$

where  $Z$  is a normalization term for images of different sizes computed as

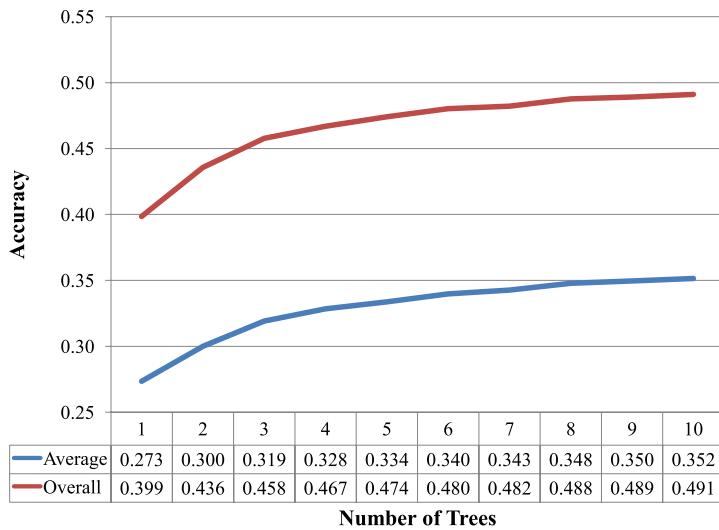
$$Z = \tilde{K}(P, P) \tilde{K}(Q, Q), \quad (7.10)$$

and  $\tilde{K}$  is the actual matching function, computed over levels of the tree as

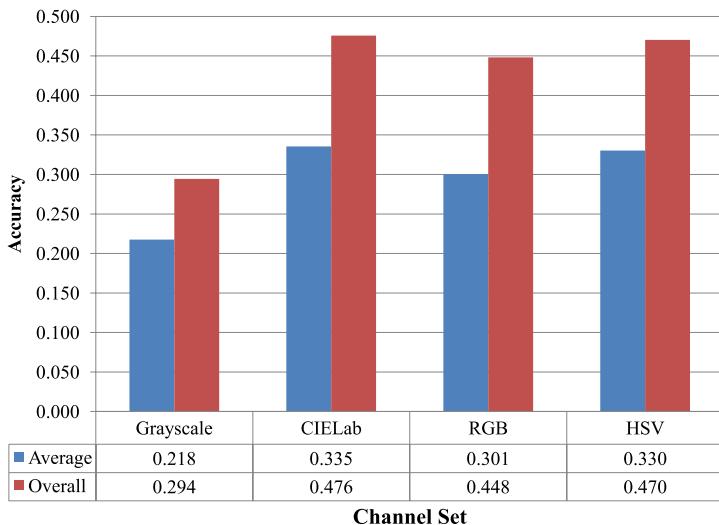
$$\tilde{K}(P, Q) = \sum_{d=1}^D \frac{1}{2^{D-d+1}} (\mathcal{I}_d - \mathcal{I}_{d+1}), \quad (7.11)$$

using the histogram intersection  $\mathcal{I}$  [33]

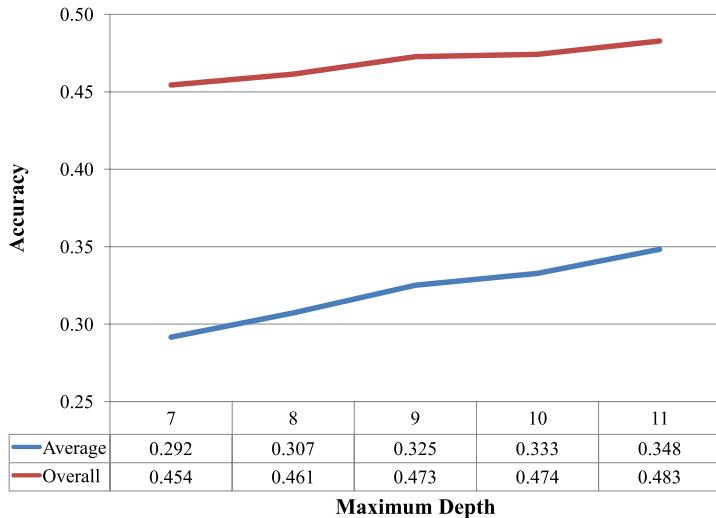
$$\mathcal{I}_d = \sum_j \min(P_d[j], Q_d[j]), \quad (7.12)$$



**Fig. 7.7** Effect of Increasing the Number of Trees. The values shown are the mean over ten trials of the overall per-pixel accuracy ( $\alpha$ ) and the mean category accuracy ( $\mu$ ). The computations for  $\alpha$  and  $\mu$  are described in Section 7.3.2. We see here a clear logarithmic growth in performance with forest size, with the elbow of the graph occurring at 5 for this dataset.



**Fig. 7.8** Effect of Different Channels. The values shown are the mean over ten trials of the overall per-pixel accuracy ( $\alpha$ ) and the mean category accuracy ( $\mu$ ). The computations for  $\alpha$  and  $\mu$  are described in Section 7.3.2. There is a slight but consistent advantage to be gained by using orthogonal color spaces over RGB, and the addition of color gives a significant improvement over grayscale, as is to be expected.



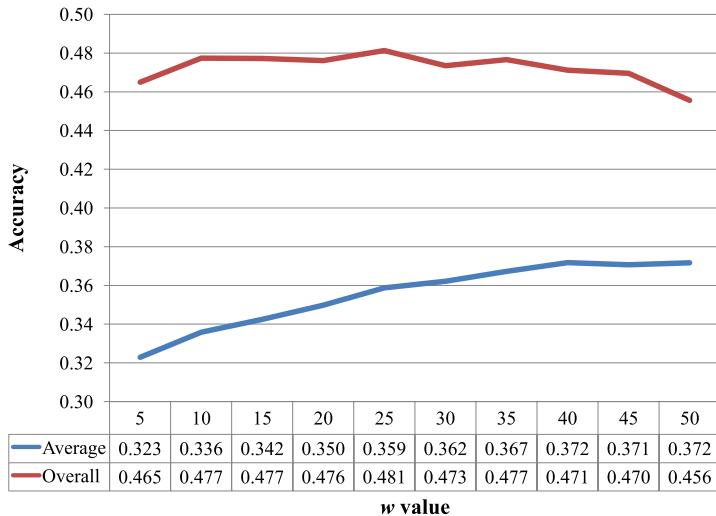
**Fig. 7.9** Effect of Maximum Depth. The values shown are the mean over ten trials of the overall per-pixel accuracy ( $\alpha$ ) and the mean category accuracy ( $\mu$ ). The computations for  $\alpha$  and  $\mu$  are described in Section 7.3.2. We see here that tree depth results in a linear growth in accuracy, particularly for class average accuracy. While the discriminative power of the forest increases with maximum depth, so does the possibility of overfitting to the data. This can be avoided by only training on subsets of the data, but one is faced with the problem of having enough data to fill the tree so as to model the correct uncertainty (as the amount of data required increases exponentially with each additional level)

where  $D$  is the depth of the tree,  $P$  and  $Q$  are BOSTs, and  $P_d$  and  $Q_d$  are the portions of the histograms at depth  $d$ , with  $j$  indexing over all nodes at depth  $d$ . There are no nodes at depth  $D + 1$ , hence  $\mathcal{I}_{D+1} = 0$ . If the tree is not full depth, missing nodes  $j$  are simply assigned  $P_d[j] = Q_d[j] = 0$ .

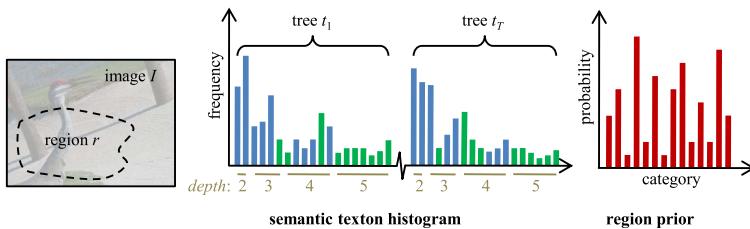
#### 7.4.2 Categorization Results

In this experiment, a forest was trained using the following parameters, selected based on the results of the experiments in Section 7.3.2.

Parameter	Value
Number of Trees	5
Maximum Depth	10
Feature Tests	$A, A + B, A \log(B),  A - B $ , Rectangle, $A - B$
Pool Size	600
w	10
Channels	CIELab
Data %	25

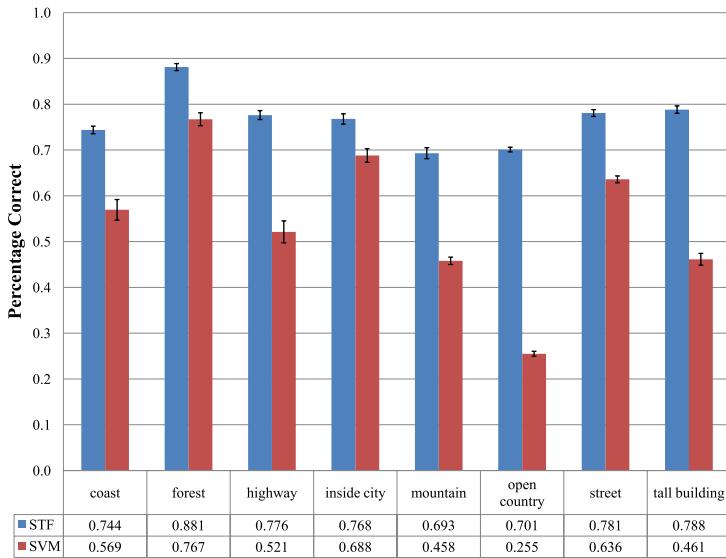


**Fig. 7.10** Effect of  $w$ . The values shown are the mean over ten trials of the overall per-pixel accuracy ( $\alpha$ ) and the mean category accuracy ( $\mu$ ). The computations for  $\alpha$  and  $\mu$  are described in Section 7.3.2. We see here a general trend by which as the parameter  $w$  increases class average accuracy increases, but overall accuracy declines. Further experiments beyond the value of 55 were inconclusive due to the effect of less data being available to the training process, but seem to support this trend.



**Fig. 7.11** Bags of semantic textons. Within a region  $r$  of image  $I$  we generate the semantic texton histogram and region prior. The histogram incorporates the implicit hierarchy of clusters in the STF, containing both STF leaf nodes (green) and split nodes (blue). The depth  $d$  of the nodes in the STF is shown. The STFs need not be to full depth, and empty bins in the histogram are not shown as the histogram is stored sparsely. The region prior is computed as the average of the individual leaf node category distributions  $P(c|l)$ .

The experiments were performed on Oliva and Torralba's scene recognition dataset from [23], in order compare the performance of SVM classifiers trained using semantic texton forests and a standard bag of words method (using Lowe's SIFT detector and descriptor [16], a vector quantized dictionary and a radial basis function-based kernel). As can be seen in Figure 7.12, semantic texton forests achieve better performance than the naive bag of words technique. What is worth

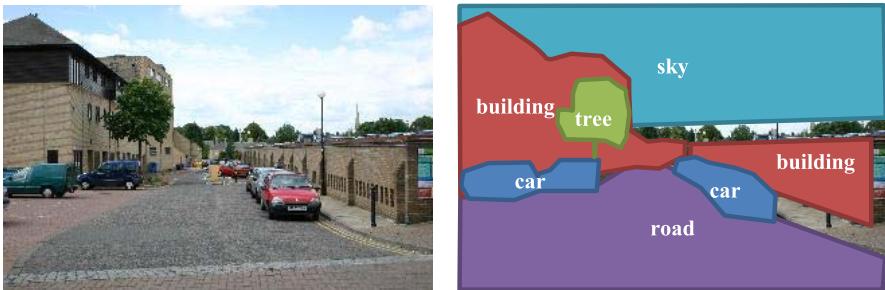


**Fig. 7.12** Scene Categorization Results. The values shown are image accuracy per category. An SVM was trained using BOSTs, and its performance was compared against an SVM trained using a standard bags-of-words model using Oliva and Torralba’s scene recognition dataset from [23]. As can be seen, the technique is on par, if not slightly better in some cases, than the bag of words based algorithm.

mentioning is that STFs are able to combine many different cues in the image aside from interest points, yet are able to compute the BOST for an image very efficiently. Due to the complexity of the process ( $O(n \log n)$ ) it is feasible for this system to perform categorization at frame rate, which would be difficult to achieve for many bag of words models.

## 7.5 Semantic Segmentation

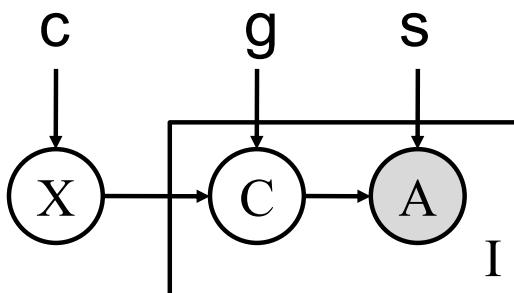
The idea of the semantic segmentation of an image is built on a model of image generation which is based on dividing the real-valued and continuous visual signal of an image into *cells*, or a regular rectangular sample grid, each of which is sufficiently explained by an underlying label. To perform a semantic segmentation of an image is to infer the semantic label for every cell. For example, look at the image in Figure 7.13. Using simple semantic labels, the pixels in the image have been explained, each one generated by some unknown model for the category label. If such a segmentation can be achieved, then the image can be catalogued for image search, used for navigation, or any number of other tasks which require basic semantic understanding of arbitrary scenes.



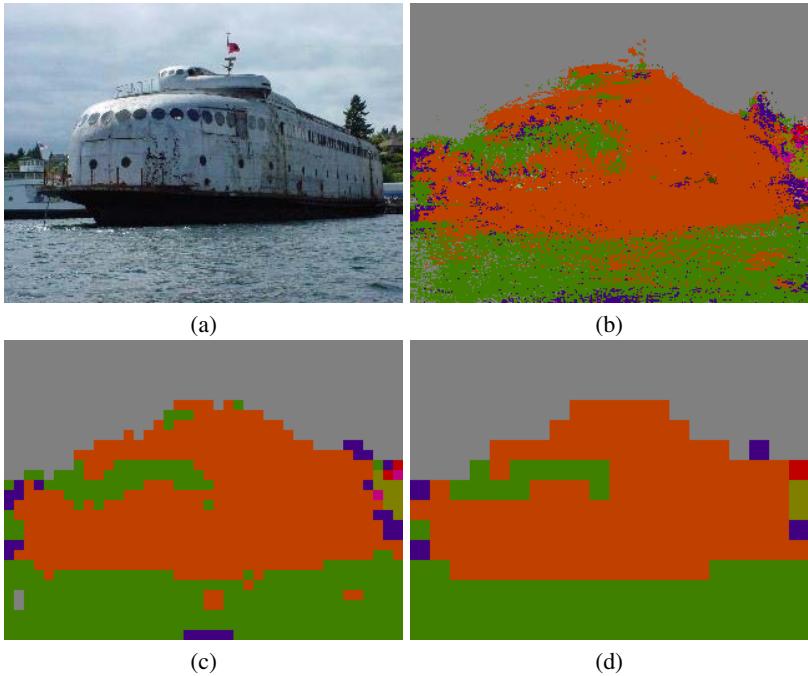
**Fig. 7.13** Semantic Segmentation. A semantic segmentation of an image is one which groups the pixels together by common semantic meaning. Shown is one such segmentation of an image, using as pixel labels the objects in the scene.

## 7.6 Soft Classification of Pixels

Our formulation of semantic segmentation centers on the cell model, presented graphically in Figure 7.14 (graphical models are a common method of visualizing joint distributions over several random variables, see [2]). In this model, for every image there is a random variable  $X$  whose value represents the subject matter, or broad image-level category, of the image (e.g., the forest, an office, the moon). The likelihood of the various image-level categories are governed by some learned parameter  $\chi$ . This topic generates cell labels  $c_i$  for each grid cell  $i$ . These are general semantic categories, that is categories of object or entity (e.g., trees, computers, rocks). The conditional probability of a semantic category given the image-level category is governed by the learned parameter  $\gamma$ . Finally, we have the appearance of a cell (e.g., a pixel patch, a descriptor, a histogram) which is generated by the



**Fig. 7.14** Cell-based Image Generation Model. This figure is a graphical representation of the model [2]. For each image a particular topic is chosen, represented by the random variable  $x$ . These can be thought of as scenes (e.g., the forest, an office, the moon). For a particular topic, we generate  $I$  cells on a grid, where each cell has a semantic category  $c$  that generates the appearance  $a$ . To perform a semantic segmentation, we infer the semantic category for each grid cell.



**Fig. 7.15** Cell Segmentation. (a) is the original image. (b) is the image with pixel-level maximum *a posteriori* (MAP) labeling. In (c), the distributions are subsampled from (b) by 8, and in (d) the image in (b) has been subsampled by 16. As grid square size increases, detail is lost but the overall accuracy of the segmentation increases.

cell category using some process which adds in some noise (indicated by  $\sigma$ ), e.g., a normal distribution over intensity values.

Naturally, a normal distribution over intensity values is not a sufficient generative patch model. While a general patch model may be out of reach in the near future, the promising jigsaw model [14] and other emerging techniques utilizing deep inference are showing great promise towards achieving this enviable goal. In the meantime, however, the best results have been obtained via discriminative methods. Since the appearance is observed, we can infer the category label from the appearance (essentially, reverse the arrow between  $c$  and  $A$ ) and infer  $c$  by marginalizing over the topics and incorporating a discriminative model for  $P(c|A)$ . Previously we discussed semantic texton forests and their ability to estimate a distribution over a set of labels for arbitrary pixel regions in a discriminative manner. Now we will discuss how they can be used to infer  $P(c|A)$  at the level of an image grid cell.

We have already established that we can infer  $P(c|A)$  for cells at the level of a pixel with a semantic texton forest:

$$P(c|A_p) = P(c|L(p)) \propto \sum_{t=1}^T P(c|l_t)P(t) . \quad (7.13)$$

As the size of a cell increases, we are faced with the problem of agglomerating the information held in individual pixels to regions. We can calculate this as

$$P(c|A_r) = P(c|r) \propto \sum_p P(c|L(p))P(p|r), \quad (7.14)$$

which leaves the conditional distribution  $P(p|r)$  to be modeled. There are two practical choices for this. The first is a bivariate normal distribution with diagonal covariance and centered on the cell, and the second is

$$P(p|r) = \begin{cases} \frac{1}{|P_r|} & p \in P_r \\ 0 & \text{otherwise} \end{cases} \quad (7.15)$$

where  $P_r$  is the set of pixels in a region. We use the latter method here due to the easy of computation, but the former likely results in a better estimate. Figure 7.15 depicts both the pixel- and cell-level maximum *a posteriori* labelings for an image.

So far, we have discussed the conditional distribution  $P(c|A)$ , but we have yet to touch on  $P(c|X)$  or  $P(X)$ . In the following section, we will discuss how it is possible to again use semantic texton forests to infer the parameter  $\chi$  from the image data.

## 7.7 Image-Level Semantic Constraints

The inference of  $P(X)$  is essentially the task of image categorization. The task of categorizing an image consists of determining those categories (e.g., forest images, office images, moon images) to which an image belongs. There has been much research performed on this problem, with the most successful of previous approaches using global image information [24], bags of words [9] or textons [39]. The STF categorization algorithm can be extended to exploit not just the hierarchy of semantic textons but also the node prior distributions  $P(c|n)$ . A non-linear support vector machine (SVM) is still used, which depends on a kernel function  $K$  that defines the similarity measure between images. To take advantage of the hierarchy in the STF, we adapt the innovative pyramid match kernel [11] to act on a pair of BOST histograms computed across the whole image, computed in the same way as described in Section 7.4.

The kernel over all trees in the STF is calculated as  $K = \sum_t \kappa_t K_t$  with mixture weights  $\kappa_t$ . Similarly to [40],  $\kappa_t = \frac{1}{T}$  results in the best categorization results. This method is very effective, but can be improved by using the learned distributions  $P(c|n)$  in the STF. If  $P(c|n)$  is large for class  $c$  and node  $n$ , then a large count of node  $n$  is a strong indicator that class  $c$  is present in the image. If  $P(c|n)$  is small, less information is gained since the likely presence of other categories only helps as context. For example, if the target of a search is grass in an image, the count of nodes likely to be grass is more important than those likely to be motorbike. Following this intuition, a 1-vs-others SVM kernel  $K_c$  is built per category, in which the count for node  $n$  in the BOST histogram is weighted by the value  $P(c|n)$ . This helps balance the categories, by selectively down-weighting those that cover large

image areas (e.g., grass, water) and thus have inappropriately strong influence on the pyramid match, masking the signal of smaller classes (e.g., cat, bird).

In these experiments, the improvement that the pyramid match kernel on the hierarchy of semantic textons gives over a radial basis function on histograms of just leaf nodes is demonstrated. An improvement using the per-category kernels  $K_c$  instead of a global kernel  $K$  is also shown.

### 7.7.1 Categorization Results

The mean average precisions (AP) in Table 7.3 compare the modified pyramid match kernel (PMK) to a radial basis function (RBF) kernel, and compare the global kernel  $K$  to the per-category kernels  $K_c$ . In the baseline results with the RBF kernel, only the leaf nodes of the STF are used, separately per tree, using term frequency/inverse document frequency to normalize the histogram. The PMK results use the entire BOST which for the per-category kernels  $K_c$  are weighted by the prior node distributions  $P(c|n)$ . Note that the mean AP is a much harder metric and gives lower numbers than recall precision or AuC; the best result in the table shows very accurate categorization. As can be seen in Table 7.3, the pyramid match kernel considerably improves on the RBF kernel. By training a per-category kernel, a small but noticeable improvement is obtained. Due to its performance, the PMK with per-category kernels to train the SVM is used as  $\chi$ .

**Table 7.3** Image categorization results. (Mean AP).

	Global kernel $K$	Per-category kernel $K_c$
RBF	.499	.525
PMK	.763	<b>.783</b>

### 7.7.2 The Image Level Prior

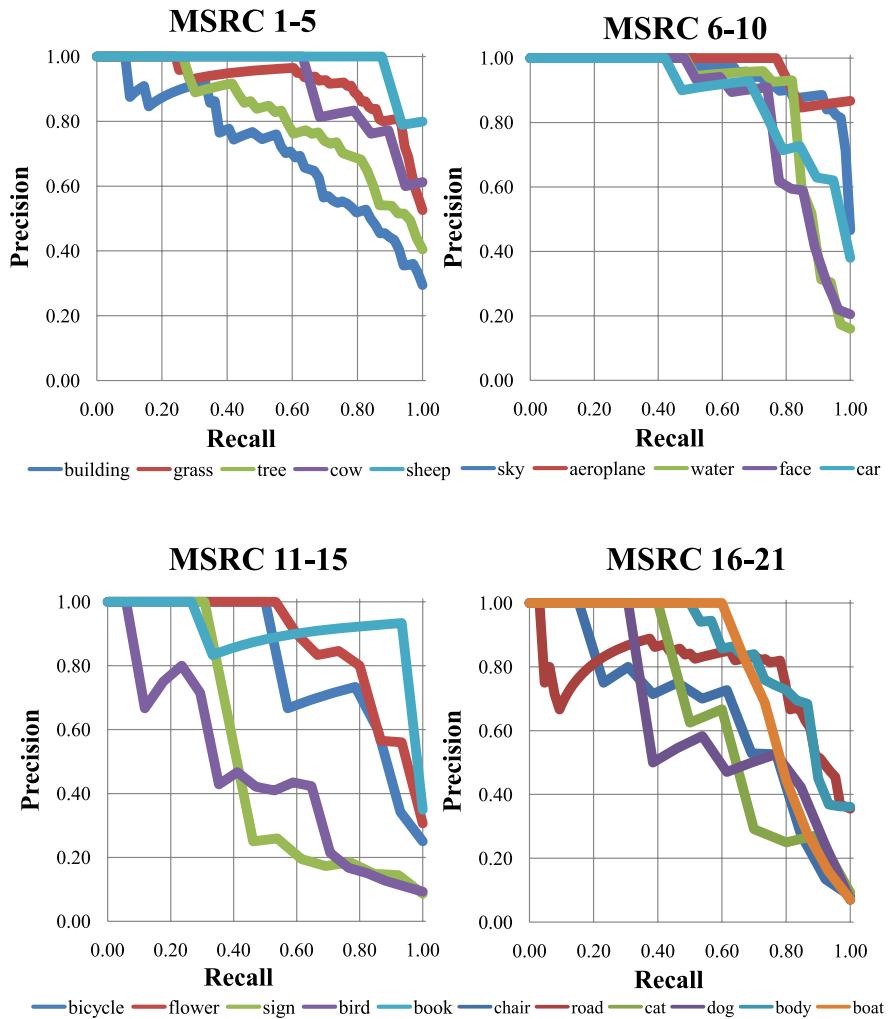
To relate this system of image categorization back to the cell model for semantic segmentation, the task the SVM is performing is to infer the value of  $X$  for an image. In essence, instead of  $\chi$  being a global prior distribution over topics, it is this system, and depends on the BOST computed for a particular image in order to compute  $P(X)$ . We can think of  $P(X)$  as being related to  $\chi$  thus:

$$P(X) = \chi(\text{BOST}_d)[x] \quad (7.16)$$

where  $\chi$  is the SVM classifier.

Moreover, since  $X$  and  $c$  in the scenario just presented have the same domain,  $\gamma$  is not a learned distribution but instead a binary function:

$$\gamma[t, c] = \begin{cases} 1 & \text{if } x = c \\ 0 & \text{otherwise} \end{cases} \quad (7.17)$$



**Fig. 7.16** MSRC Categorization Results Shown here are the precision recall curves for all of the categories in the MSRC21 dataset.

Thus, whereas one would usually marginalize over  $X$  when computing  $P(c_i)$  in the following manner (substituting lower case letters for observed variables):

$$P(c_i) = P(c_i|a_i) \sum_X P(X) P(c_i|X) \prod_{j \neq i} \sum_{c_j} P(c_j|X) P(c_j|a_j) \quad (7.18)$$

this equation is derived instead :

$$P(c_i) = P(c_i|a_i) P(c_i|x) P(x) \prod_{j \neq i} \sum_{c_j} P(c_j|x) P(c_j|a_j) \quad (7.19)$$

which, given the binary nature of  $P(c|T) = \lambda[t, c]$  and that in this situation  $t \equiv c$ , collapses further to:

$$P(c_i) = P(c_i|a_i)P(c_i) \prod_{j \neq i} P(c_j|a_j) \quad (7.20)$$

where  $P(c_i)$  is the result of the SVM classifier and  $P(c|A)$  is computed using the STF as described above. Furthermore, the effect of the right-hand product is estimated by a power  $\alpha$  on  $P(c)$  in the results presented in this chapter. Since this is no longer inference *per se*, the result of this process is referred to as an “image level prior”, or ILP. While these optimizations result in a system which is very efficient, they are made at the cost of a more powerful and expressive model.

## 7.8 Compositional Constraints

To demonstrate the power of the BOSTs as features for segmentation, they are integrated into the TextronBoost algorithm [30]. The goal is to segment an image into coherent regions and simultaneously infer the class label of each region. In [30], a boosting algorithm selected features based on localized counts of textons to model patterns of texture, layout and context. The context modeled in [30] was “textural”, for example: sheep often stand on something green. We adapt the rectangle count features of [30] to act on both the semantic texton histograms and the BOST region priors. The addition of region priors allows us to model context based on *semantics* [26], not just texture. Continuing the example, this new model can capture the notion that sheep often stand on grass.

The segmentation algorithm works as follows. For speed, a second randomized decision forest is used in place of boosting. This *segmentation forest* is trained to act at image cells  $i$ , using bags and priors computed using Equation 7.15 for  $P(p|r = i)$ . At test time, the segmentation forest is applied at each pixel  $p$  densely or, for more speed, on a grid. The most likely class in the averaged category distribution gives the final segmentation for each cell. The split tests compute either the count  $H_{r+i}(n = n')$  of semantic texton  $n'$ , or the probability  $P(c | r + i)$  of class  $c$ , within rectangle  $r$  translated relative to cell  $i$ . By translating rectangle  $r$  relative to the cell  $i$  being classified, and by allowing  $r$  to be a large distance away from  $i$  (up to half the image size), such features can exploit texture, layout and context information. This extension to their features exploits semantic context by using the region prior probabilities  $P(c|r + i)$  inferred by the semantic textons.

## 7.9 Experiments

Before presenting in-depth results for segmentation, let us look briefly at the STFs themselves. In Figure 7.17, we visualize the inferred leaf nodes  $L = (l_1, \dots, l_T)$  for each pixel  $i$  and the most likely category  $c_i = \arg \max_{c_i} P(c_i|L)$ . Observe that

**Fig. 7.17** Textonizations. Shown here are a large selection of textonizations performed by a semantic texton forest. The first column shows the image, the middle five are textonizations from the five trees in the forest, followed by the ground truth image and the combined textonization of the forest. In the textonizations, a separate color is given to each texton index to give a sense of leaf-membership for a pixel.

the textons in each tree capture different aspects of the underlying texture and that even at such a low level the distribution  $P(c|L)$  contains significant semantic information. Table 7.4 gives a naïve segmentation baseline on the MSRC dataset by comparing  $c_i$  to the ground truth.

Clearly, this segmentation is poor, especially when trained in a weakly supervised manner, since only very local appearance and no context is used. Even so, the signal is remarkably strong for such simple features (random chance is under 5%). Below

**Table 7.4** Naïve Segmentation Baseline on MSRC21. Using the parameters chosen as a result of the experiments in Section 7.3.2 we are able to obtain a solid baseline for segmentation.

	Global Average	
supervised	48.7%	41.5%
weakly supervised	17.7%	27.8%

is shown how using semantic textons as features in higher level classifiers greatly improves these numbers, even with weakly supervised or unsupervised STFs.

Except where otherwise stated, STFs were used with the following parameters, hand-optimized on the MSRC validation set: distance  $w = 31$ ,  $T = 5$  trees, maximum depth  $D = 10$ , 500 feature tests and 5 threshold tests per split, and  $\frac{1}{4}$  of the data per tree, resulting in approximately 500 leaves per tree. Training the STF on the MSRC dataset took only 15 minutes. The tests used were  $A + B$ ,  $A - B$ ,  $|A - B|$ , and  $A$ , a combination motivated by the experiments in Section 7.3.2 and performance requirements.

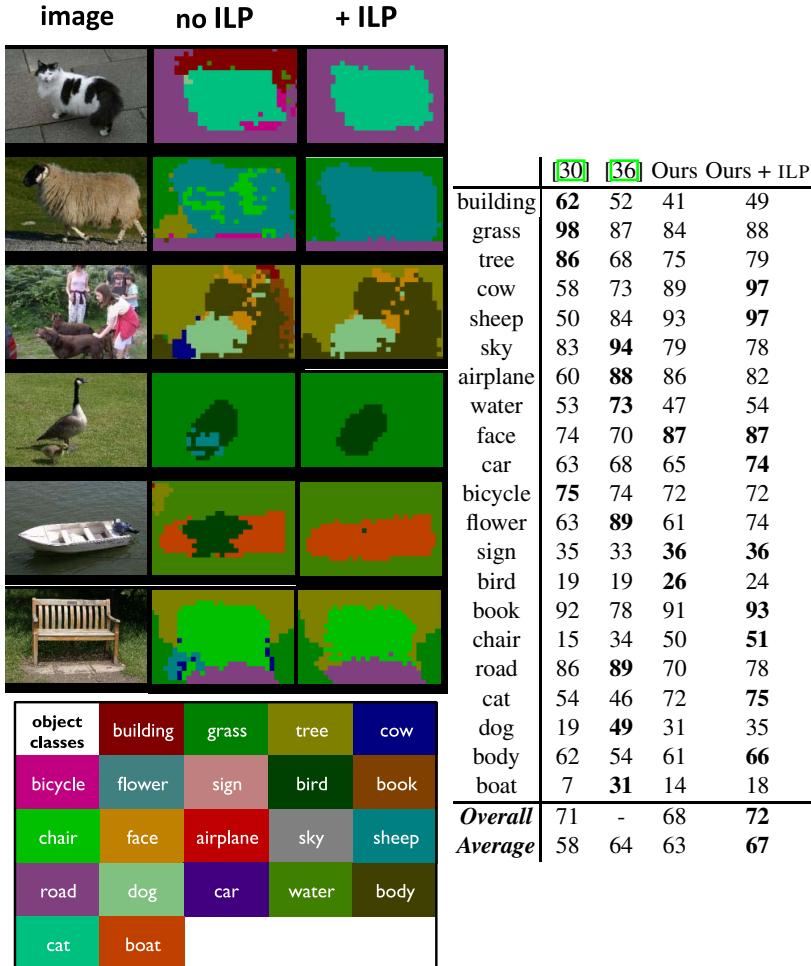
### 7.9.1 MSRC21 Dataset

We first examine the influence of different aspects of our system on segmentation accuracy. Segmentation forests were trained using (a) the histogram  $H_r(l)$  of just leaf nodes  $l$ , (b) the histogram  $H_r(n)$  of all tree nodes  $n$ , (c) just the region priors  $P(c|r)$ , (d) the full model using all nodes and region priors, (e) the full model trained without random transformations, (f) all nodes using an unsupervised STF (no region priors are available), and (g) all nodes using a weakly-supervised STF (only image labels). The category average accuracies are given in Table 7.5 with and without the image-level prior.

There are several conclusions to draw. (1) In all cases the ILP improves results. (2) The hierarchy of clusters in the STF gives a noticeable improvement. (3) The region priors alone perform remarkably well. Comparing to the segmentation result using only the STF leaf distributions (34.5%) this shows the power of the localized BOSTs that exploit semantic context. (4) Each aspect of the BOST adds to the model. While, without the ILP, score (b) is slightly better than the full model (d), adding

**Table 7.5** Comparative segmentation results on MSRC.

	Without ILP	With ILP
(a) only leaves	61.3%	64.1%
(b) all nodes	<b>63.5%</b>	65.5%
(c) only region priors	62.1%	66.1%
(d) full model	63.4%	<b>66.9%</b>
(e) no transformations	60.4%	64.4%
(f) unsupervised STF	59.5%	64.2%
(g) weakly supervised STF	61.6%	64.6%



**Fig. 7.18** MSRC21 segmentation results. Left: Segmentations on test images using semantic texton forests. Note how the good but somewhat noisy segmentations are cleaned up using our image-level prior (ILP) that emphasizes the categories likely to be present. (Note that neither a Markov nor conditional random field are used, which could clean up the segmentations to precisely follow image edges [30]). Right: Segmentation accuracies (percent) over the whole dataset, without and with the ILP. Highly efficient semantic textons achieve a significant improvement on previous work.

in the ILP shows how the region priors and textons work together<sup>1</sup> (5) Random transformations of the training images improve performance by adding invariance.

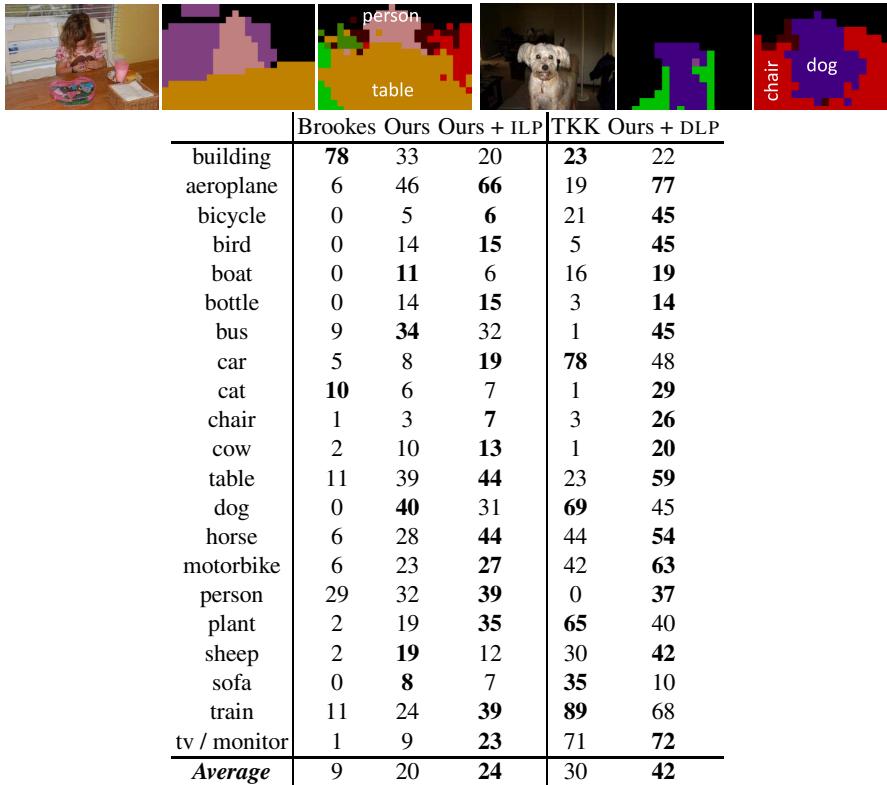
<sup>1</sup> This effect may be due to segmentation forest (b) being over-confident: looking at the 5 most likely classes inferred for each pixel, (b) achieves 87.6% while (d) achieves a better 88.0%.



**Fig. 7.19** Further MSRC segmentation results.

(6) Performance increases with more supervision, but even unsupervised STFs allow good segmentations.

Given this insight, the algorithm is compared against [30] and [36]. The same train/test split is used as in [30] (though not [36]). The results are summarized in Figure 7.18, with further segmentation results in Figure 7.19. Across the whole challenging dataset, using the full model with ILP achieved a class average performance of 66.9%, a significant improvement on both the 57.7% of [30] and the 64% of [36]. The global accuracy also improves slightly on [30]. The image-level prior improves



	Brookes	Ours	Ours + ILP	TKK	Ours + DLP
building	<b>78</b>	33	20	<b>23</b>	22
aeroplane	6	46	<b>66</b>	19	<b>77</b>
bicycle	0	5	<b>6</b>	21	<b>45</b>
bird	0	14	<b>15</b>	5	<b>45</b>
boat	0	<b>11</b>	6	16	<b>19</b>
bottle	0	14	<b>15</b>	3	<b>14</b>
bus	9	<b>34</b>	32	1	<b>45</b>
car	5	8	<b>19</b>	<b>78</b>	48
cat	<b>10</b>	6	7	1	<b>29</b>
chair	1	3	<b>7</b>	3	<b>26</b>
cow	2	10	<b>13</b>	1	<b>20</b>
table	11	39	<b>44</b>	23	<b>59</b>
dog	0	<b>40</b>	31	<b>69</b>	45
horse	6	28	<b>44</b>	44	<b>54</b>
motorbike	6	23	<b>27</b>	42	<b>63</b>
person	29	32	<b>39</b>	0	<b>37</b>
plant	2	19	<b>35</b>	<b>65</b>	40
sheep	2	<b>19</b>	12	30	<b>42</b>
sofa	0	<b>8</b>	7	<b>35</b>	10
train	11	24	<b>39</b>	<b>89</b>	68
tv / monitor	1	9	<b>23</b>	71	<b>72</b>
<i>Average</i>	9	20	<b>24</b>	30	<b>42</b>

**Fig. 7.20** VOC 2007 segmentation results. Above: Test images with ground truth and our inferred segmentations using the ILP (not the DLP). This dataset is extremely challenging and the resulting segmentations are thus slightly noisier. Below: Segmentation accuracies (percent) over the whole dataset. The left three results compare the method to the Brookes segmentation entry [8], and show that it is over twice as accurate. The two results on the right compare the best automatic segmentation-by-detection entry (see text) [8] with the STF algorithm using the TKK results as a detection-level prior (DLP). The algorithm improves the accuracy of segmentation-by-detection by over 10%.

performance for all but three classes, but even without it, results are still highly competitive with other methods. The use of balanced training has resulted in more consistent performance across classes, and significant improvements for certain difficult classes: cow, sheep, bird, chair, and cat. A Markov or conditional random field is not used here, which would likely further improve our performance [30].

These results used the learned and extremely fast STFs, without needing any slow hand-designed filter-banks or descriptors. Extracting the semantic textons at every pixel takes an average of only 275 milliseconds per image, categorization takes 190 ms, and evaluating the segmentation forest only 140 ms. For comparison [30] took over 6 seconds per test image, and [36] took an average of over 2 seconds per image for feature extraction and between 0.3 to 2 seconds for estimating the segmentation.

This algorithm is well over 5 times faster *and* improves quantitative results. A real-time implementation of the STFs can be achieved on a standard PC, as the complexity is just  $O(TD)$  per pixel.

The following parameter settings were used:  $T = 50$  trees of depth  $D = 14$ , with training examples taken every 10 pixels in a random 50% of the training images, and using 1000 random feature tests and 20 random threshold tests at each split node. The ILP smoothing term  $\alpha = 0.5$ . To add invariance, the training set was augmented: the original plus 3 copies with random transformations of rotation up to 6°, scaling up to 1.2x, left-right flipping, and affine intensity changes up to  $1.2I + 0.05$ . Training took under 2 hours for the segmentation forest. The resulting forest used a total of 24805 node features and 107311 region prior features.

### 7.9.2 VOC 2007 Segmentation Dataset

The VOC object recognition challenge added a segmentation task to the competition in 2007 [8]. This dataset contains 21 extremely challenging categories including background. A STF, a segmentation forest, and an ILP were trained on this data, using the “trainval” split and keeping parameters as for MSRC. The results in Figure 7.20 compare with [8]. This algorithm performs over twice as well as the only *segmentation* entry (Brookes), and the addition of the ILP further improves performance by 4%. The actual winner of the segmentation challenge, the TKK algorithm, used *segmentation-by-detection* that fills in the detected object bounding boxes by category. To see if our algorithm could use a *detection*-level prior DLP (identical to the ILP but using the detected bounding boxes and varying with image position) the TKK entry output was used as the DLP. The STF algorithm gave a large 12% improvement over the TKK segmentation-by-detection, highlighting the power of STFs as features for segmentation.

## 7.10 Discussion

We have examined semantic texton forests and their applications to image categorization and semantic segmentation. They act as efficient texton codebooks, which do not depend on local descriptors or expensive  $k$ -means clustering, and when supervised during training can infer a distribution over categories at each pixel. We examined how the training parameters affect performance, and how a hierarchical matching kernel can be used in a non-linear SVM classifier to achieve image categorization. Also, we saw how bags of semantic textons enabled state-of-the-art performance on challenging datasets for semantic segmentation, and how the use of an inferred image-level prior significantly improves segmentation results. The substantial gains of the method over traditional texton-based methods are training and testing efficiency and improved quantitative performance.

While semantic texton forests are quite powerful, there is much work left to be done. One limitation of the system is the large dimensionality of the bag of semantic textons. This necessitates a trade-off between the memory usage of the semantic

texton integral images and the training time if they are computed at runtime. While using just the region priors can be more memory efficient, this comes at some cost in pixel accuracy. Another weakness, mentioned previously, is the lack of a model of the inter-pixel relationships (like the conditional Markov random field from [30]) which incorporates an understanding of local structure in the image when arriving at a segmentation.

Perhaps the greatest limitation of this technique, however, is the fact that it is a discriminative learning algorithm and thus requires a fully-supervised training scenario. As the data required must be labeled on a per-pixel basis with a preset number of categories, this is a large barrier to the general use of semantic texton forests for image understanding. However, as the genesis of a generative patch model nears, it is possible that a hybridized version of the technique could be used in partially supervised and unsupervised cases. Perhaps most encouraging are the vast quantities of training data being created through projects like LabelMe [27], in which humans provide semantic segmentations of images of all kinds with arbitrary categories. While difficult to work with, such data could be exploited by discriminative learning algorithms like the semantic texton forest to understand a wider range of object categories.

## References

1. Amit, Y., Geman, D.: Shape quantization and recognition with randomized trees. *Neural Computation* 9(7), 1545–1588 (1997)
2. Bishop, C.: *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc. (2006)
3. Bosch, A., Zissermann, A., Muñoz, X.: Image classification using random forests and ferns. In: *Proceedings of the International Conference on Computer Vision* (2007)
4. Breiman, L.: Random forests. *Machine Learning* 45(1), 5–32 (2001)
5. Breiman, L., Friedman, J., Olshen, R.: *Classification and Regression Trees*. Wadsworth, Belmont (1984)
6. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: *Proceedings of the International Workshop on Statistical Learning in Computer Vision, ECCV* (2004)
7. Elkan, C.: Using the triangle inequality to accelerate  $k$ -means. In: *Proceedings of the International Conference on Machine Learning*, pp. 147–153 (2003)
8. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL VOC Challenge (2007),  
<http://www.pascal-network.org/challenges/VOC/voc2007/index.html>
9. Fei-Fei, L., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2005)
10. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. *Machine Learning* 36(1), 3–42 (2006)
11. Grauman, K., Darrell, T.: The pyramid match kernel: Discriminative classification with sets of image features. In: *Proceedings of the International Conference on Computer Vision* (2005)

12. Jain, A.K.: Fundamentals of Digital Image Processing. Prentice-Hall, New Jersey (1989)
13. Jurie, F., Triggs, B.: Creating efficient codebooks for visual recognition. In: Proceedings of the International Conference on Computer Vision, pp. 604–610 (2005)
14. Lasserre, J., Kannan, A., Winn, J.: Hybrid learning of large jigsaws. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, Minneapolis (2007)
15. Lepetit, V., Lagger, P., Fua, P.: Randomized trees for real-time keypoint recognition. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 775–781 (2005)
16. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
17. Malik, J., Belongie, S., Leung, T., Shi, J.: Contour and texture analysis for image segmentation. International Journal of Computer Vision 43(1), 7–27 (2001)
18. Marée, R., Geurts, P., Piater, J., Wehenkel, L.: Random subwindows for robust image classification. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 34–40 (2005)
19. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. International Journal of Computer Vision 60(1), 63–86 (2004)
20. Moosmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomized clustering forests. In: Proceedings of the International Conference on Neural Information Processing Systems (2006)
21. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2006)
22. Nowak, E., Jurie, F., Triggs, B.: Sampling strategies for bag-of-features image classification. In: Proceedings of the International Conference on Computer Vision (2006)
23. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. International Journal of Computer Vision 42(3), 145–175 (2001)
24. Oliva, A., Torralba, A.: Building the gist of a scene: The role of global image features in recognition. Visual Perception, Progress in Brain Research 155(1), 23–26 (2006)
25. Quelhas, P., Monay, F., Odobez, J.M., Gatica, D., Tuytelaars, T.: Modeling scenes with local descriptors and latent aspects. In: Proceedings of the International Conference on Computer Vision (2005)
26. Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., Belongie, S.: Objects in context. In: Proceedings of the International Conference on Computer Vision (2007)
27. Russell, B., Torralba, A., Murphy, K., Freeman, W.T.: Labelme: a database and web-based tool for image annotation. Journal of Computer Vision 77(1-3), 157–173 (2008)
28. Russell, B.C., Efros, A.A., Sivic, J., Freeman, W.T., Zisserman, A.: Using multiple segmentations to discover objects and their extent in image collections. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2006)
29. Schindler, G., Brown, M., Szeliski, R.: City-scale location recognition. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, Minneapolis (2007)
30. Shotton, J., Winn, J., Rother, C., Criminisi, A.: Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. International Journal of Computer Vision 81(1) (2009)
31. Sivic, J., Russel, B., Efros, A., Zisserman, A., Freeman, W.: Discovering objects and their localization in images. In: Proceedings of the International Conference on Computer Vision, Beijing, China, pp. 370–377 (2005)
32. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: Proceedings of the International Conference on Computer Vision, vol. 2, pp. 1470–1477 (2003)

33. Swain, M., Ballard, D.: Color indexing. *Int. J. Computer Vision* 7, 11–32 (1991)
34. Tuytelaars, T., Schmid, C.: Vector quantizing feature space with a regular lattice. In: *Proceedings of the International Conference on Computer Vision* (2007)
35. Varma, M., Zisserman, A.: A statistical approach to texture classification from single images. *International Journal of Computer Vision* 62(1-2), 61–81 (2005)
36. Verbeek, J., Triggs, B.: Region classification with markov field aspect models. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2007)
37. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 511–518 (2001)
38. Winder, S., Brown, M.: Learning local image descriptors. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2007)
39. Winn, J., Criminisi, A., Minka, T.: Object categorization by learned universal visual dictionary. In: *Proceedings of the International Conference on Computer Vision*, Beijing, China, pp. 1800–1807 (2005)
40. Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision* 73(2), 213–238 (2007)

# Chapter 8

## Multi-view Object Categorization and Pose Estimation

Silvio Savarese and Li Fei-Fei

**Abstract.** Object and scene categorization has been a central topic of computer vision research in recent years. The problem is a highly challenging one. A single object may show tremendous variability in appearance and structure under various photometric and geometric conditions. In addition, members of the same class may differ from each other due to various degrees of intra-class variability. Recently, researchers have proposed new models towards the goal of: i) finding a suitable representation that can efficiently capture the intrinsic three-dimensional and multi-view nature of object categories; ii) taking advantage of this representation to help the recognition and categorization task. In this Chapter we will review recent approaches aimed at tackling this challenging problem and focus on the work by Savarese & Fei-Fei [54, 55]. In [54, 55] multi-view object models are obtained by linking together diagnostic parts of the objects from different viewing point. Instead of recovering a full 3D geometry, parts are connected through their mutual homographic transformation. The resulting model is a compact summarization of both the appearance and geometry information of the object class. We show that such a model can be learnt via minimal supervision compared to competitive techniques. The model can be used to detect objects under arbitrary and/or unseen poses by means of a two-step algorithm. This algorithm, inspired by works in single object view synthesis (e.g., Seitz & Dyer [57]), has the ability to synthesize object appearance and shape properties at recognition time, and in turn estimate the object pose that best matches the observations. We conclude this Chapter by presenting experiments on detection, recognition and pose estimation results with respect to two datasets in [54, 55] as well as to PASCAL Visual Object Classes (VOC) dataset [15]. Experiments indicate that representation and algorithms presented in [54, 55] can be successfully employed in a number of generic object recognition tasks.

---

Silvio Savarese

University of Michigan at Ann Arbor, Department of Electrical Engineering, USA  
e-mail: [silvio@umich.edu](mailto:silvio@umich.edu)

Li Fei-Fei

Stanford University, Department of Computer Science USA  
e-mail: [feifeili@cs.stanford.edu](mailto:feifeili@cs.stanford.edu)



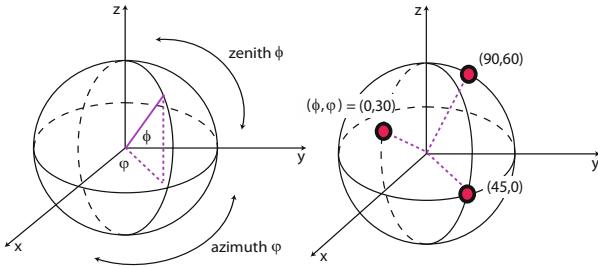
**Fig. 8.1** Categorize an Object Given An Unseen View. azimuth: [front,right,back,left]=[0, 90, 180, 270] $^o$ ; zenith: [low, med., high]=[0, 45, 90] $^o$

## 8.1 Introduction

The ability to interpret a scene, recognize the objects within, estimate their location and pose is crucial for a robust, intelligent visual recognition system. In robotic manipulation, a robotic arm may need to detect and grasps objects in the scene such as a cup or book; in autonomous navigation, an unmanned vehicle may need to recognize and interpret the behavior of pedestrians and other vehicles in the environment. Critically, accurate pose recovery is not only important if one wants to interact with the objects in the environment (if a robotic arms wishes to grasp a mug, the system must estimate mug's pose with high degree of accuracy); it is also a key ingredient that enables the visual system to perform higher level tasks such activity or action recognition. Despite recent successful efforts in the vision community toward the goal of designing systems for object recognition, a number of challenges still remain: not only does one need to cope with traditional nuisances in object categorization problems (objects appearance variability due to intra-class changes, occlusions and lighting conditions), but also to handle view-point variability and propose representations that capture the intrinsic multi-view nature of object categories.

In this Chapter we describe a recent recognition paradigm for discovering object semantics under arbitrary viewing conditions as well as recovering the basic geometrical attributes of object categories and their relationships with the observer. Figure 8.1 illustrates more precisely the problem we would like to solve. Given an image containing some object(s), we would like to learn object category models that allow us to: i) detect and categorize the object as a car (or a stapler, or a computer mouse), and ii) estimate the pose (or view point) of the car. Here by ‘pose’, we refer to the 3D information of the object that is defined by the viewing angle and scale of the object (i.e., a particular point on the viewing sphere represented in Figure 8.2).

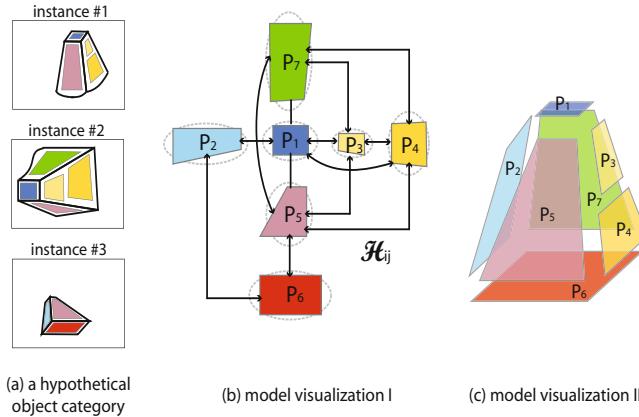
Most of the recent advances in object categorization have focused on modeling the appearance and shape variability of objects viewed from a limited number of poses [66, 19, 18, 35, 17, 23], or a mixture of poses [56, 67, 48, 72]. In these methods, different poses of the same object category result in completely independent models, wherein neither features nor parts are shared across views. These methods typically ignore the problem of recovering the object pose altogether. We refer to such models



**Fig. 8.2** Left: An object pose is represented by a pair of azimuth and zenith angles. Right: Some of the unseen poses tested during our recognition experiments (Figure 8.12).

as *single-view 2D models*. At the opposite end of the spectrum, several works have addressed the issue of single object recognition by modeling different degree of 3D information [40, 50, 6, 20]. Since these methods achieve recognition by matching local features under rigid geometrical transformations, they are successful in recovering the object pose, but they are difficult to extend to handle object classes. We refer to such models as *single instance 3D models*. Similar limitations are suffered by representations based on aspect graphs [31, 32, 59, 58, 13, 47, 5, 12, 10]. A small but growing number of recent studies have begun to address the problem of object classification in a true multi-view setting [63, 33, 27, 70, 9, 54, 55, 37, 49, 71, 2, 16, 61, 60, 43]. Since in these methods object elements (features, parts, contours) are connected across views so as to form an unique and coherent model for the object category (e.g., Figure 8.4), we refer to such models as *multi-view models*. These techniques bridge the gap between single view 2D models and single instance 3D object models. In this book Chapter we will focus on the framework introduced by [54, 55], which represents one of the first attempts to model the multi-view nature of 3D object categories. In particular we will discuss some of the critical contributions of such multi-view model [54, 55]:

- A part-based 3D model of an object class is proposed by encoding both the appearance and 3D geometric shape information (see Figure 8.3). Stable parts of objects from one class are linked together to capture both the appearance and shape properties of the object class. This model produces a compact yet power representation of an object class, differing from most of the previous works which store various image exemplars or model exemplars of different viewing angles.
- Toward the goal of learning the multi-view model of an object class, the algorithm demands less supervision in the learning process compared to previous works (i.e., [63]). The method is designed to handle either segmented or unsegmented objects in training. Most importantly, the method does not require view point labels or to have training images sorted in any particular order. A key assumption, however, is that multiple views of the same object instance are assumed to be available.



**Fig. 8.3** Schematic illustration of the 3D object category model. (a) We illustrate our model by using a hypothetical 3D object category. Three instances of the hypothetical objects are shown here as sample training images. The colored regions of each instance are “canonical parts” of the objects that will be put together to form the final model. These “canonical parts” are made of patches usually provided by feature detectors (see also Figure 8.4). When parts across different instances share the same color code, it indicates that the model has learnt the correspondences among these regions based on the appearance and geometric consistency. (b) The final model of the 3D object category can be viewed as a connected graph of the object canonical parts (colored regions,  $P_i$ ), canonical part relations ( $\mathcal{H}$ ), and the encoded variabilities of the appearances and geometric structure (visualized by the dashed ellipses surrounding each part). (c) A more intuitive visualization of the model puts together the canonical parts in a 3D graph based of the learned geometric relations ( $\mathcal{H}$ ). This figure is best viewed under color.

- The algorithm has the ability to represent and synthesize views of object classes that are not present in training. The view-synthesis approach is inspired by previous research on view morphing and image synthesis from multiple views. The main contribution of [54, 55] is that the synthesis takes place at the categorical level as opposed to the single object level (as previously explored).
- Given a novel testing image containing an object class, not only does the algorithm classify the object, but it also infers the pose and scale and localizes the object in the image. Furthermore, the algorithm takes advantage of our view-synthesis machinery for recognizing objects seen under arbitrary views. As opposed to [9] where training views are augmented by using synthetic data, we synthesize the views at *recognition* time.
- Extensive experimental validation is provided. Competitive categorization, localization and pose estimation performances are observed with respect to the dataset in [63] as well as the challenging 3D object datasets introduced in [54, 55].

## 8.2 Literature Review

Interest in solving the 3D/multi-view object recognition as well as pose estimation problem starts with a number of seminal work [3][24][39][41][46][51][65] in the 80s and early 90s, which form the foundation of modern object recognition. Beginning from the late 90s, researchers start proposing a new generation of techniques for solving single object recognition using *single instance 3D models*. In [38][42][44], objects are represented by highly discriminative and local invariant features related by local geometric constraints. Methods by [50][6][20] follow the idea of enforcing global geometric constraints and/or achieve 3D affine or Euclidean object reconstruction from multiple (often) unregistered views for recognizing single objects in cluttered scenes. These methods are successful thanks to their ability to identify strong geometrical constraints and highly discriminative features. However, such constraints are not adequate in object categorization problems in which shape and appearance variability of each object class must be accounted for. Another large body of literature on object recognition introduces the concept of *Aspect Graph* (AG). AGs represent 3D objects as a set of topologically distinct views based on visibility constraints. Starting from seminal works of [31][32], different AG representations are introduced during the 80s and 90s [59][58][13][47][5][14] until recent extensions [12][10]. Similarly to single instance 3D models, AG methods lack of generalization power in representing object categories, and have shown limited success in modeling intra-class variability. Also, most of AGs poorly handle nuisances such as occlusions and background clutter. A natural step forward to the categorization problem is offered by *3D object category classification methods* such as [52][28][26][22][30][62][36]. These methods focus on classifying objects that are expressed as collections of 3D points, 3D meshes or 3D synthetic cad models. Often 3D shape databases are used [1]. Due to their limited ability to coherently integrate real-world object albedo information with the underlying 3D structure, these methods are hardly used to identify real world objects in cluttered scenes from still images or videos. A recent survey summarizes relevant literature [62]. Partially to accommodate intra-class variability, researchers have proposed to leverage on the large literature on single 2D view object categorization and represent 3D object categories as a *mixture of 2D single view object category models* [56][67][7][64][4]. In mixture models, single view object models are completely independent, wherein neither features or parts are linked across views. An exception is the work by [64] where an efficient multi-class boosting procedure is introduced to limit the computational overload. The consequence is that mixture models fail to form a coherent and compact multi-view representation of an object category. Methods based on mixture models are costly to train and prone to false alarm, if several views need to be encoded. Finally, only few methods [67] have attempted to solve the pose estimation problem as we will discuss later in more details.

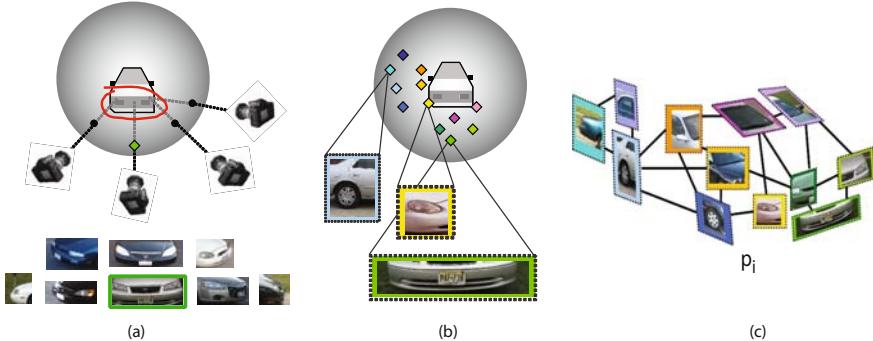
Recently, a new class of methods have tried to bridge the gap between single view 2D models and single instance 3D object models, and have begun to address the problem of object classification in a true multi-view setting (*multi-view models*). In these methods, object elements (features, parts, contours) are connected

across views so as to form an unique and coherent model for the object category (e.g., Figure 8.4). Pioneering multi-view techniques are introduced in the specific domain of face detection [48, 72]. Methods in [63, 33] extend single view models into the multi-view setting by linking relevant features across views. Alternative techniques [27, 70, 37, 69] represent an object category by using synthetic or reconstructed 3D models on top of which the typical distribution of appearance element is learned. Authors in [9] build an object representation upon a 3D skeleton model of predefined parts from 2D images. Very recent examples of multi-view representations are [61, 60, 49, 71, 2, 16]. The work by [54, 55] proposes a new representation where object categories are modeled as a collection of view point invariant parts connected by relative view point transformations. [54, 55] stand among the pioneering contributions on multi-view representation and are among the first methods that have addressed the issue of view point estimation for generic object categories. In the remainder of this book Chapter we discuss in details the representation introduced in [54, 55]. In Section 8.3.2 we explain the multi-view model based on canonical parts and linkage structure. Then we describe how to learn such multi-view model in Section 8.4. In Section 8.5 we present the machinery for synthesizing novel views in the viewing sphere. In Section 8.6 we demonstrate how a novel instance of an object category is recognized, localized and its pose inferred. Finally, we show experimental results in Section 8.7.

## 8.3 The Part-Based Multi-view Model

### 8.3.1 Overview

In [54, 55] models of an object category are obtained by linking together diagnostic parts (also called canonical parts) of the objects from different viewing points. As previous research has demonstrated, a part-based representation [34] is more stable for capturing appearance variability of object categories across instances and views. Canonical parts are discriminative and view invariant representations of local planar regions attached to the object physical surface. Such parts are modeled by distributions of vector quantized features [11]. Instead of expressing part relationships by recovering the full 3D geometry of the object, [50, 6, 20], canonical parts are connected through their mutual homographic transformations and positions. The resulting model is a compact summarization of both the appearance and geometrical information of the object categories across views (rather than being just a collection of single view models). Effectively, the linkage structure can be interpreted as the generalization to the multi-view case of single 2D constellation or pictorial structure models [68, 19, 18] where parts or features are connected by a mere 2D translational relationship. [54, 55]'s framework requires less supervision than competing techniques ([63, 33, 27, 70, 37, 69]) (where often pose labels are required). Similarly to other constellation methods, [54, 55]'s model enables a recognition system that is robust to occlusions and background clutter. Finally, and most importantly, by introducing the view morphing constraints, [55] has demonstrated the ability to



**Fig. 8.4** Canonical parts and linkage structure. (a): A car within the viewing sphere. As the observer moves on the viewing sphere the same part produces different appearances. The location on the viewing sphere where the part is viewed the most frontally gives rise to a canonical part. The appearance of such canonical part is highlighted in green. (b): Colored markers indicate locations of other canonical parts. (c): Canonical parts are connected together in a linkage structure (see also Figure 8.3). The linkage indicates the relative position and change of pose of a canonical part given the other (if they are both visible at the same time). This change of location and pose is represented by a translation vector and a homographic transformation respectively. The homographic transformation between canonical parts is illustrated by showing that some canonical parts are slanted with respect to others. A collection of canonical parts that share the same view defines a canonical view (for instance, see the canonical parts enclosed in the dashed rectangle).

predict appearance and location of parts that are not necessarily canonical. This is useful for recognizing objects observed from arbitrary viewing conditions (that is, from views that are not seen in learning) and critical for improving the false alarm rate (a consequence of single view object representations). [54,55]'s framework for recognizing poses of generic object categories is among the earliest attempts of this kind (along with [48,72,67]).

### 8.3.2 Canonical Parts and Linkage Structure

In this Section we describe in details the concept of canonical parts and linkage structures. The central ideas are summarized in Figure 8.3 and Figure 8.4. They offer a schematic view of the core components of the model through a hypothetical 3D object category.

The **appearance** information is captured in the diagnostic parts of the objects in one class, denoted as  $P_i$ . Each “part” is a region of an object that tends to appear consistently throughout different instances of the same category (shown in colored patches in Figure 8.3(a)). It is a collection of a number of smaller image patches usually provided by the feature detectors, constraint by some geometric consistency.

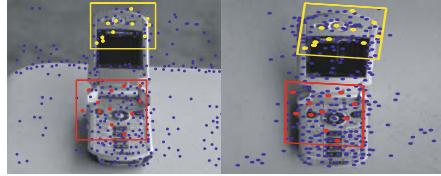
Readers familiar with the current object recognition literature are reminded that our “part” is not a single detected region such as Harris corner or DoG detection, but rather a larger structure that contains many detected local regions. Given  $P_i$ , our model also encodes the appearance variations observed in training in the form of distributions of descriptors. In our model, we call such diagnostic parts *canonical parts* as they are representative of parts viewed in their most frontal position. For example, the canonical part representation of the car rear bumper is the one that is viewed the most frontally (Figure 8.3(b) and 8.4(a)).

Given an assortment of canonical parts (e.g., the colored patches in Figure 8.4(b)), a *linkage structure* connects each pair of canonical parts  $\{P_j, P_i\}$  if they can be both visible at the same time (Figure 8.3(b) and 8.4(c)). The linkage captures the relative position (represented by the  $2 \times 1$  vector  $\mathbf{t}_{ij}$ ) and change of pose of a canonical part given the other (represented by a  $2 \times 2$  homographic transformation  $\mathcal{A}_{ij}$ ). If the two canonical parts share the same pose, then the linkage is simply the translation vector  $\mathbf{t}_{ij}$  (since  $\mathcal{A}_{ij} = \mathbf{I}$ ). For example, given that part  $P_i$  (left rear light) is canonical, the pose (and appearance) of all connected canonical parts must change according to the transformation imposed by  $\mathcal{A}_{ij}$  for  $j = 1 \dots N, j \neq i$ , where  $N$  is the total number of parts connected to  $P_i$ . This transformation is depicted in Figure 8.4(c) by showing a slanted version of each canonical part.

We define a *canonical view*  $V$  as the collection of canonical parts that share the same view  $V$  (Figure 8.4(c)). Thus, each pair of canonical parts  $\{P_i, P_j\}$  within  $V$  is connected by  $\mathcal{A}_{ij} = \mathbf{I}$  and a translation vector  $\mathbf{t}_{ij}$ . We can interpret a canonical view  $V$  as a subset of the overall linkage structure of the object category. Notice that by construction a canonical view may coincide with one of the object category poses used in learning. However, not all the poses used in learning will be associated to a canonical view  $V$ . The reason is that a canonical view is a collection of canonical parts and each canonical part summarizes the appearance variability of an object category part under different poses. The relationship of parts within the same canonical view is what previous literature have extensively used for representing 2D object categories from single 2D views (e.g., the constellation models [68, 19]). The linkage structure can be interpreted as its generalization to the multi-view case. Similarly to other methods based on constellations of features or parts, the linkage structure of canonical parts is robust to occlusions and background clutter.

## 8.4 Building the Model

We detail here the algorithm for building a 3D object class model from a set of training images. We assume that each training image contains one instance of the target object class. We do not, however, have information about the instance membership or pose of the object. The task of learning is to start with this set of raw images, extract features to form parts, obtain a set of *canonical parts* and finally form the object class model by connecting these canonical parts across views.



**Fig. 8.5** Detected features using the scaled invariant saliency detector [29]. All interest points are indicated by blue dots. The boxed regions in each image denote the learnt parts for this pair. When two parts across images share the same color (i.e., red boxes), they are connected by the algorithm. This figure should be viewed in color.

#### 8.4.1 Extract Features

Local image patches are the basic building blocks of an object image. The algorithm, however, works independently of any particular choice of feature detectors or descriptors [44][38]. In practice, we choose the Saliency detector [29] and the SIFT descriptor [38] to characterize local features. An image  $i$  therefore contains hundreds of detected patches, each represented as  $f_i = (\mathbf{a}_i, \mathbf{x}_i)$ , where  $\mathbf{a}_i$  is the appearance of the patch, described by a 128-dimension SIFT vector, and  $\mathbf{x}_i$  is the location of the feature on the 2D image. Figure 8.5 shows two examples of cellphone images and their detected patches.

#### 8.4.2 Form Parts

The 3D object category model is represented in a hierarchical way. Local image features are first grouped into larger regions (called “parts”). A selected subset of these parts (according to appearance and geometric consistency) are then linked together as a full 3D model. This choice stems from the observation that larger regions of objects often carry more discriminative information in appearance and are more stable in their geometric relationships with other parts of the object [34].

The goal of this step is to group local image features into “parts” that are consistent in appearance and geometry across images. A global geometrical constraint is obtained by imposing that feature match candidates (belonging to different views) are related by the fundamental matrix  $\mathcal{F}$ . A local geometrical constraint is enforced by imposing that features belonging to a neighborhood are related by homographic transformation  $\mathcal{H}$  induced by  $\mathcal{F}$  [25]. We use a scheme based on RANSAC [21] to enforce such constraints while the optimal  $\mathcal{F}$  and  $\mathcal{H}$  are estimated. Below is a brief sketch of the algorithm.

1. Obtain a set of  $M$  candidate features based on appearance similarity measured by  $d(\mathbf{a}_i - \mathbf{a}_j)$  across 2 training images.
2. Run RANSAC algorithm on  $M$  to obtain a new (and smaller) set of matches  $M_F \in M$  based on  $\mathbf{x}_i \mathcal{F} \mathbf{x}_j = 0$ , where  $\mathcal{F}$  denotes the fundamental matrix.

3. Further refine the matches using RANSAC to obtain a set of  $M_H$  matches such that  $\mathbf{x}_i - \mathcal{H}\mathbf{x}_j = 0$ , where  $M_H \in M_F \in M$ .

Step 2 and Step 3 can be iterated until the residual error computed on the inliers stops decreasing. Step 3 returns a pair of local neighborhood regions across the 2 training images in which all features  $f_i \in M_H^{(i,j)}$  satisfy a vicinity constraint. We call them a matched “part”. We follow this procedure for every pair of training images. Figure 8.5 shows example parts indicated by boxes on these two cellphone images. Note that there is no presumed shape or size of these parts.

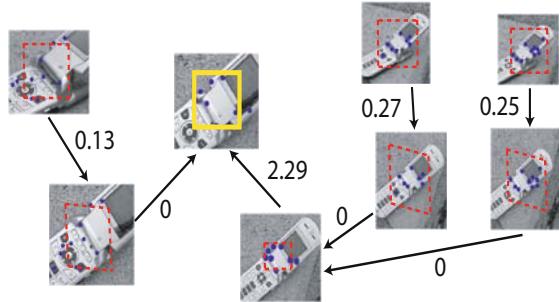
**Implementation Details.** On average parts contain 50 – 200 features, sufficient to effectively represent the local structure of the object from a particular view. We obtain on average 700 – 1000 matched parts within a training set of 48 images. We use a mask to remove spurious matches coming from the background. This is not a requirement for the algorithm to work. 6 shows that spurious matches can be effectively removed by enforcing global constraints across all the views. Finally, even if matched parts can be obtained from pairs of images belonging to different instances of a given category, we have noticed that the algorithm in 8.4.2 mostly produces matched parts from images belonging to the same object instance. This is due to the inherent lack of flexibility of RANSAC to handle intra-class variability. In fact, this is an advantage because it guarantees robustness and stability in the part matching process. Actual matching of corresponding parts belonging to different object instances is achieved in the optimization process detailed in 8.4.4.

#### 8.4.2.1 Representing Canonical Parts

Each canonical part is represented by a distribution of feature descriptors along with their  $x, y$  location within the part. Specifically, we describe a canonical part  $P$  by a convex quadrangle  $B$  (e.g., the bounding box) enclosing the set of features. The appearance of this part is then characterized by a *bag of codewords* model 11 – that is, a normalized histogram  $h$  of vector quantized descriptors contained in  $B$ . A standard K-means algorithm can be used for extracting the codewords.  $B$  is a  $2 \times 4$  vector encoding the  $b = [x, y]^T$  coordinates of the four corners of the quadrangle, i.e.,  $B = [b_1 \dots b_4]$ ;  $h$  is a  $M \times 1$  vector, where  $M$  is the size of the vocabulary of the vector quantized descriptors. Given a linked pair of canonical parts  $\{P_i, P_j\}$  and their corresponding  $\{B_i, B_j\}$ , relative position of the parts  $\{P_i, P_j\}$  is defined by  $\mathbf{t}_{ij} = c_i - c_j$ , where the centroid  $c_i = \frac{1}{4} \sum_k b_k$ ; the relative change of pose is defined by  $A_{ij}$  which encodes the homographic transformation acting on the coordinates of  $B_i$ . This simplification is crucial for allowing more flexibility in handling the synthesis of novel non-canonical views at the categorical level.

#### 8.4.3 Find Canonical Parts Candidates

Our goal is to represent the final object category with “canonical parts” and their mutual geometric relations. To do so, we need to first propose a set of canonical part

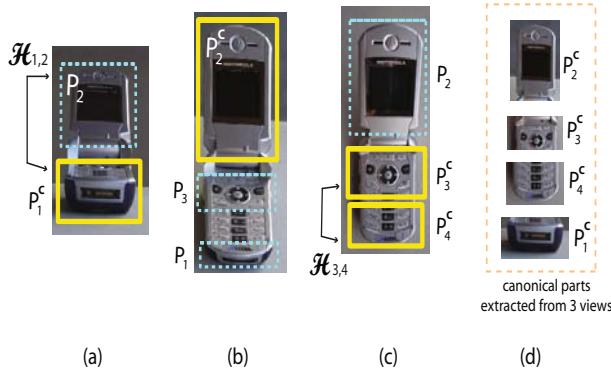


**Fig. 8.6** Illustration of linked parts for proposing one canonical part of the cellphone model using directed graph. The boxes indicate parts associated with this canonical part. The blue dots indicate detected local features within the parts. The yellow box is the proposed canonical part by summarizing all *factors of compression* (indicated by the numerical value adjacent to each arrow) given all the connected paths.

candidates based on a view-point criteria. What we have from the training images is a large set of “parts” that are paired across different images, each part consisting of a number of local features. Many of these parts linked across different images correspond to one *actual* part of the object (e.g., LCD screen of a cellphone). Figure 8.6 is a illustration of the connected parts estimated from Step 8.4.2. The most possible front view of an actual object part defines a *canonical part candidate*. This will be by definition the canonical pose attached to the canonical part candidate. A canonical part candidate can be computed from the set of linked parts as follows.

Between every connected pair of parts, we associate them with a *factor of compression* cost  $\mathcal{K}_{ij}$ .  $\mathcal{K}_{ij}$  is a function of  $\mathcal{A}_{ij}$  in the homographic relationship  $\mathcal{H}_{ij}$  between these two parts.  $\mathcal{H}_{ij}$  is provided by the algorithm in section 8.4.2. Specifically,  $\mathcal{K}_{ij} = (\lambda_1^{ij} \lambda_2^{ij} - 1)$ , where  $\lambda_{1,2}^{ij}$  are the two singular values of  $\mathcal{A}_{ij}$ .  $\mathcal{K}_{ij}$  is greater than 0 when  $P_i$  is a less compressed version than  $P_j$  under affine transformation. Using the sign of  $\mathcal{K}_{ij}$ , we assign the direction between two parts. The full set of parts and their directed connections weighted by  $\mathcal{K}_{ij}$  form a weighted directed graph (Figure 8.6). It is easy to show that the path associated to highest value of the total factor of compression cost ( $\sum_{(i,j) \in \text{path}} \mathcal{K}_{ij}$ ) gives rise to a canonical part candidate for it can be identified as the part  $P$  attached to the terminal node of such maximum cost path. The intuition here is that the maximum cost path is the one that leads to the part with smallest compression, thus the canonical one. The maximum cost path can be found with a simple greedy algorithm.

**Implementation Details.** The graph structure is on average composed of 10 – 15 parts but can go as low as 2, if a part is shared by only two views. For that reason, the greedy algorithm finds the optimal solution very quickly. Special care, however, needs to be taken if the graph contains loops. This may occur when the orientation of a part is estimated with low accuracy from the previous step. Typically the number of canonical part candidates is one third of the initial set of part candidates.



**Fig. 8.7** Illustration of the canonical parts and their geometric relations for three views of the same object. The yellow box indicates the canonical part of interest that is viewed given its canonical pose (i.e., most frontal view by definition). The examples of canonical regions extracted from these three views are shown in the box on the right. The dashed cyan boxes indicate parts that do not share the same pose with the yellow canonical part. The cyan parts have a canonical counter part in a different pose. In this example we use the symbol "c" to differentiate a canonical part from its "non-canonical" counterpart. For instance, there exists a linkage structure between canonical parts  $P_1^c$  and  $P_2^c$ . The  $\mathcal{H}_{12}$  denotes the transformation to observe  $P_2^c$  when  $P_1^c$  is viewed in its canonical position (thus, generating cyan  $P_2$ ). In the right most pose, two canonical parts  $P_3^c$  and  $P_4^c$  share the same canonical pose. In this case, the transformation  $\mathcal{H}_{34}$  is just a translation because  $P_3^c$  and  $P_4^c$  are canonical at the same time.

#### 8.4.4 Create the Model

Section 8.4.3 has proposed a number of canonical part candidates from the training images. So far, we have only utilized local appearance or pair-wise geometry information to find correspondences between parts and find the canonical part candidates. Now we are ready to take all these candidates to obtain a canonical part at the categorical level. This allows propose a 3D object category model by finding a globally consistent and optimal combination of canonical parts.

We use the same notation ( $P_i$ ) to indicate a canonical part of a given category. The context can help differentiate the categorical case from the single instance case. As anticipated in Section 8.3.2, given two different canonical part  $P_i$  and  $P_j$ , there are two ways that they are placed with respect to each other onto the 3D object model. In the first case, when  $P_i$  is viewed frontally,  $P_j$  is also viewed frontally (Figure 8.7, right panel). In this case the homographic linkage between these two canonical parts is  $\mathcal{H}_{ij} = \begin{bmatrix} \mathbf{I} & \mathbf{t}_{ij} \\ \mathbf{0} & 1 \end{bmatrix}$ , where  $\mathbf{I}$  is the identity matrix. In the second case,  $P_i$  and  $P_j$  are not viewed frontally simultaneously. They are, therefore, related by a full homographic  $\mathcal{H}_{ij} = \begin{bmatrix} \mathcal{A}_{ij} & \mathbf{t}_{ij} \\ \mathbf{0} & 1 \end{bmatrix}$ .  $\mathcal{H}_{ij}$  denotes the transformation to observe  $P_j$  when  $P_i$  is viewed in its most front view position. Parts  $P_1$  when  $P_2$  in Figure 8.7 have this type of linkage.

$\mathcal{A}_{ij}$  captures both the 2D relationship (e.g., position) between canonical parts as well as a soft 3D relationship which provided by the affinity transformation  $\mathcal{A}_{ij}$  between parts. Canonical parts that are not connected correspond to sides of the object that can never be seen at the same time. As introduced in Section 8.3.2, we define a *canonical view*  $V$  as the collection of canonical parts that share the same view  $V$  (Figure 8.4(c)). Thus, each pair of canonical parts  $\{P_i, P_j\}$  within  $V$  is connected by  $\mathcal{A}_{ij} = \mathbf{I}$  and a translation vector  $\mathbf{t}_{ij}$ .

Given the pool of candidate canonical parts from all the instances of a given category, we wish to calculate the set of canonical parts at the categorical level. This can be done by matching corresponding candidate canonical parts across all the instances. This correspondence problem can be solved by means of an optimization process that jointly minimizes the appearance difference between matching candidates and their corresponding linkage structure  $\mathcal{A}_{ij}$ .

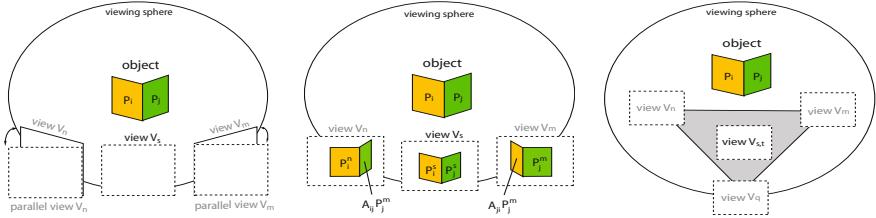
The 1<sup>st</sup> row of Figure 8.14 (2<sup>nd</sup> and 3<sup>rd</sup> columns) shows an illustration of the learnt cellphone model. The model obtained thus far provides a compact representation of object parts from all the views.

**Implementation Details.** The optimization is carried out by exploiting similarity of appearance and the estimated linkage structure between canonical part candidates belonging to different object instances. The appearance similarity is computed as a chi-square distance between the histograms representing the canonical region appearances. Similarity of linkage structure is computed by comparing  $\mathcal{A}_{ij}$  for every pairs of canonical parts candidates  $P_i, P_j$ . Notice that this optimization step greatly benefits from the fact that parts-to-be-matched are canonical. This means that all the parts are already normalized in term of their viewing angle and scale. Furthermore, the number of canonical part candidates is a small subset the initial number of parts. All this greatly simplifies the matching process which could have been hardly feasible otherwise.

## 8.5 View Synthesis

### 8.5.1 Representing an Unseen View

The critical question is: how can we represent (synthesize) a novel non-canonical view from the set of canonical views contained in the linkage structure? As we will show in Section 8.6, this ability becomes crucial if we want to recognize an object category seen under an arbitrary pose. The approach is inspired by previous research on view morphing and image synthesis from multiple views. We show that it is possible to use a similar machinery for synthesizing appearance, pose and position of canonical parts from two or more canonical views. Notice that the output of this representation (synthesis) is a novel view of the object *category*, not just a novel view of a single object instance, whereas all previous morphing techniques are used for synthesizing novel views of single objects.



**Fig. 8.8** View Synthesis. Left: If the views are in a neighborhood on the viewing sphere, the cameras can be approximated as being *parallel*, enabling a linear interpolation scheme. Middle: 2-view synthesis: A pair of linked parts  $\{P_i^s, P_j^s\} \in V^s$  is synthesized from the pair  $P_i^n \in V^n$ , and  $P_j^m \in V^m$  if and only if  $P_i^n$  and  $P_j^m$  are linked by the homographic transformation  $A_{ij} \neq I$ . Right: 3-view synthesis can take place anywhere within the triangular area defined by the 3 views.

### 8.5.1.1 View Morphing

Given two views of a 3D object it is possible to synthesize a novel view by using view-interpolating techniques without reconstructing the 3D object shape. It has been shown that a simple linear image interpolation (or appearance-morphing) between views do not convey correct 3D rigid shape transformation, unless the views are parallel (that is, the camera moves parallel to the image planes) [8]. Moreover, Seitz & Dyer [57] have shown that if the camera projection matrices are known, then a geometrical-morphing technique can be used to synthesize a new view even without having parallel views. However, estimating the camera projection matrices for the object category may be very difficult in practice. We notice that under the assumption of having the views in a neighborhood on the viewing sphere, the cameras can be approximated as being *parallel*, enabling a simple linear interpolation scheme (Figure 8.8). Next we show that by combining appearance and geometrical morphing it is possible to synthesize a novel view (meant as a collection of parts along with their linkage) from two or more canonical views.

### 8.5.1.2 Two-View Synthesis

We start by the simpler case of synthesizing from two canonical views  $V^n$  and  $V^m$ . A synthesized view  $V^s$  can be expressed as a collection of linked parts morphed from the corresponding canonical parts belonging to  $V^n$  and  $V^m$ . Specifically, a pair of linked parts  $\{P_i^s, P_j^s\} \in V^s$  can be synthesized from the pair  $\{P_i^n \in V^n, P_j^m \in V^m\}$  if and only if  $P_i^n$  and  $P_j^m$  are linked by the homographic transformation  $A_{ij} \neq I$  (Figure 8.8). If we represent  $\{P_i^s, P_j^s\}$  by the quadrangles  $\{B_i^s, B_j^s\}$  and the histograms  $\{h_i^s, h_j^s\}$  respectively, a new view is expressed by:

$$B_i^s = (1-s)B_i^n + sA_{ij}B_j^n; \quad B_j^s = sB_j^m + (1-s)A_{ji}B_i^m; \quad (8.1)$$

$$h_i^s = (1-s)h_i^n + sh_j^m; \quad h_j^s = sh_j^m + (1-s)h_i^n; \quad (8.2)$$

The relative position between  $\{P_i^s, P_j^s\}$  is represented as the difference  $\mathbf{t}_{ij}^s$  of the centroids of  $B_i^s$  and  $B_j^s$ .  $\mathbf{t}_{ij}^s$  may be synthesized as follows:

$$\mathbf{t}_{ij}^s = (1-s)\mathbf{t}_{ij}^n + s\mathbf{t}_{ij}^m \quad (8.3)$$

In summary, Equation 8.1 and 8.3 regulate the synthesis of the linkage structure between the pair  $\{P_i^s, P_j^s\}$ ; whereas Equation 8.2 regulate the synthesis of their appearance components. By synthesizing parts for all possible values of  $i$  and  $j$  we can obtain a set of linked parts which give rise to a new view  $V^s$  between the two canonical views  $V^n$  and  $V^m$ . Since all canonical parts in  $V^n$  and  $V^m$  (and their linkage structures) are represented at the categorical level, this property is inherited to the new parts  $\{P_i^s, P_j^s\}$ , thus to  $V^s$ .

### 8.5.1.3 Three-View Synthesis

One limitation of the interpolation scheme described in Section 8.5.1.2 is that a new view can be synthesized only if it belongs to the linear camera trajectory from one view to the other. By using a bi-linear interpolation we can extend this to a novel view from 3 canonical views. The synthesis can take place anywhere within the triangular area defined by the 3 views (Figure 8.8) and is regulated by two interpolating parameters  $s$  and  $t$ . Similarly to the 2-view case, 3-view synthesis can be carried out if and only if there exist 3 canonical parts  $P_i^n \in V^n$ ,  $P_j^m \in V^m$ , and  $P_k^q \in V^q$  which are pairwise linked by the homographic transformations  $\mathcal{A}_{ij} \neq I$ ,  $\mathcal{A}_{ik} \neq I$  and  $\mathcal{A}_{jk} \neq I$ . The relevant quantities can be synthesized as follows:

$$B_i^{st} = [(s-1)I \ sI] \begin{pmatrix} B_i^n & \mathcal{A}_{ik}B_i^n \\ \mathcal{A}_{ij}B_i^n & \mathcal{A}_{ik}\mathcal{A}_{ij}B_i^n \end{pmatrix} \begin{bmatrix} (1-t)I \\ tI \end{bmatrix} \quad (8.4)$$

$$h_i^{st} = [(s-1)I \ sI] \begin{pmatrix} h_i^n & h_i^q \\ h_i^m & h_i^p \end{pmatrix} \begin{bmatrix} (1-t)I \\ tI \end{bmatrix} \quad (8.5)$$

$$\mathbf{t}_{ij}^{st} = [(s-1)I \ sI] \begin{pmatrix} \mathbf{t}_{ij}^n & \mathbf{t}_{ik}^q \\ \mathbf{t}_{ij}^m & \mathbf{t}_{ij}^m + \mathbf{t}_{ik}^q - \mathbf{t}_{ij}^n \end{pmatrix} \begin{bmatrix} (1-t)I \\ tI \end{bmatrix} \quad (8.6)$$

Analogous equations can be written for the remaining indexes.

## 8.6 Recognizing Object Class in Unseen Views

Section 8.5.1 has outlined all the critical ingredients of the model for representing and synthesizing new views. We discuss here an algorithm for recognizing pose and categorical membership of a query object seen under arbitrary view point. We consider a two-step recognition procedure. Similarly to the training procedure, we first extract image features and use these to propose candidate canonical parts. This provides the input for the first step of the algorithm whose output is a short list of

*Algorithm step 1*

```

1.  $I \leftarrow$  list of parts extracted from test image
2. for each model  $C$ 
3.   for each canonical view  $V \in C$ 
4.      $[R(n), V^*(n)] \leftarrow \text{MatchView}(V, C, I);$  % return similarity  $R$ 
5.      $n ++;$ 
6.    $L \leftarrow \text{KMinIndex}(R)$  % return shortlist  $L$ 

```

$\text{MatchView}(V, C, I)$

```

1. for each canonical part  $P \in V$ 
2.    $M(p) \leftarrow \text{MatchKPart}(P, I);$  % return K best matches
3.    $p ++;$ 
4. for each canonical part  $\bar{P} \in C$  linked to  $V$ 
5.    $\bar{M}(q) \leftarrow \text{MatchKPart}(\bar{P}, I);$  % return K best matches
6.    $q ++;$ 
7.    $[M^*, \bar{M}^*] \leftarrow \text{Optimize}(V, M, \bar{M});$ 
8.    $V^* \leftarrow \text{GenerateTestView}(M^*, \bar{M}^*, I);$ 
9.    $R \leftarrow \text{Distance}(V, V^*);$ 
10. Return  $R, V^*;$ 

```

**Fig. 8.9** Pseudocode of the step 1 algorithm.  $\text{MatchView}(V, C, I)$  returns the similarity score between  $V$  and  $I$ .  $\text{KminIndex}()$  returns pointers to the the  $K$  smallest values of the input list.  $\text{MatchKPart}(P, I)$  returns the best  $K$  candidate matches between  $P$  and  $I$ . A match is computed by taking into account the appearance similarity  $S_a$  between two parts.  $S_a$  is computed as the distance between the histograms of vector quantized features contained in the corresponding part's quadrangles  $B$ .  $\text{Optimize}(V, M, \bar{M})$  optimizes over all the matches and returns the best set of matches  $M^*, \bar{M}^*$  from the candidate matches in  $M, \bar{M}$ . The selection is carried out by jointly minimizing the overall appearance similarity  $S_a$  (computed over the candidate matches) and the geometrical similarity  $S_g$  (computed over pairs of candidate matches).  $S_g$  is computed by measuring the distance between the relative positions  $\mathbf{t}_{ij}, \bar{\mathbf{t}}_{ij}$ .  $\text{GenerateTestView}(M^*, \bar{M}^*, I)$  returns a linkage structure of parts ( $B$ , appearances  $h$  and relative positions  $\mathbf{t}$ ) given  $M^*, \bar{M}^*$ . This gives rise to the estimated matched view  $V^*$  in the test image.  $\text{Distance}(V_i, V_j)$  returns an estimate of the overall combined appearance and geometrical similarity  $S_a + S_g$  between the linkage structures associated to  $V_i, V_j$ .  $S_a$  is computed as in  $\text{MatchKPart}$  over all the parts.  $S_g$  is computed as the geometric distortion between the two corresponding linkage structures.

the  $K$  best model views across all views and all categories. The second step refines the error scores of the short list by using the view-synthesis scheme.

### 8.6.1 Extract Features and Get Part Candidates

We follow the same procedure as in learning to find local interest points by using the Saliency detector [29]. Each detected patch is then characterized by a 128-dimension SIFT vector [38]. Given an object model, say the “cellphone” model,

*Algorithm step 2*

```

1. for each canonical view  $V \in L$ 
2.    $V^* \leftarrow L(l)$ 
3.    $V' \leftarrow \text{FindClosestView}(V, C);$ 
4.    $V'' \leftarrow \text{FindSecondClosestView}(V, C);$ 
5.   for each 2-view synthesis parameter  $s$ 
6.      $V^s \leftarrow \text{2-ViewSynthesis}(V, V', s);$ 
7.      $R(s) \leftarrow \text{Distance}(V^s, V^*);$ 
8.   for each 3-view synthesis parameters  $s$  and  $t$ 
9.      $V^{s,t} \leftarrow \text{3-ViewSynthesis}(V, V', V'', s, t);$ 
10.     $R(s, t) \leftarrow \text{Distance}(V^{s,t}, V^*);$ 
11.    $L(l) \leftarrow \text{Min}(R);$ 
12.    $l++;$ 
13.  $[C_w V_w] \leftarrow \text{MinIndex}(L);$ 

```

**Fig. 8.10** Pseudocode of the step 2 algorithm.  $\text{FindClosestView}(V, C)$  ( $\text{FindSecondClosestView}(V, C)$ ) returns the closest (second closest) canonical pose on the viewing sphere.  $\text{2-ViewSynthesis}(V, V', s)$  returns a synthesized view between the two views  $V, V'$  based on the interpolating parameters  $s$ .  $\text{3-ViewSynthesis}(V, V', s, t)$  is the equivalent function for three view synthesis.  $C_w$  and  $V_w$  are the winning categories and poses respectively.

we first find a list of canonical part candidates by the following procedure. For each canonical part of the model, we greedily search through the test image by a scanning window across pixel locations, scales and orientations. Canonical parts and test parts are matched by comparing the distributions of features belonging to the relevant regions. The most probably  $N$  firings (typically 5) are retained as the  $N$  candidates for a canonical part  $P_i$ . This provides hypotheses of canonical parts consistent with a certain canonical view of an object model.

### 8.6.2 Recognition Procedure: First Step

In the first step (Figure 8.9), we want to match the query image with the best object class model and pose. Given hypotheses of canonical parts consistent with a certain canonical view of an object model, we infer the appearance, pose and position of other parts that are not seen in their canonical view (MatchView function). This information is encoded in the object class linkage structure. An optimization process finds the best combination of hypothesis over appearance and geometrical similarity (*Optimize*). The optimization process is very similar to the one introduced in learning when different constellations of canonical parts are matched. The output is a similarity score as well as a set of matched parts and their linkage structure (the estimated matched view  $V^*$ ) in the test image. The operation is repeated for all possible canonical views and for all object class models. Finally, we create a short list of the  $N$  best canonical views across all the model categories ranked according

to their similarity (error) scores. Each canonical view is associated to its own class model label. The complexity of step-1 is  $O(N^2N_vN_c)$ , where  $N$  is the total number of canonical parts (typically, 200 – 500);  $N_v$  = number of views per model;  $N_c$  = number of models.

### 8.6.3 Recognition Procedure: Second Step

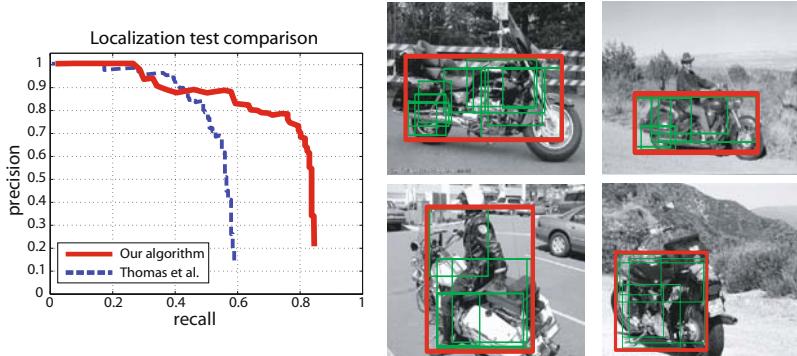
In the second step (Figure 8.10), we use the view synthesis scheme (Section 8.5.1) to select the final winning category and pose from the short list. The idea is to consider a canonical view from the short list, pick up the nearest (or two nearest) canonical pose(s) on the corresponding model viewing sphere (*FindClosestView* and *FindSecondClosestView*), and synthesize the intermediate views according to the 2-view-synthesis (or 3-view-synthesis) procedure for a number of values of  $s$  ( $s, t$ ) (*2-ViewSynthesis* and *3-ViewSynthesis*). For each synthesized view, the similarity score is recomputed and the minimum value is retained. We repeat this procedure for each canonical view in the short list. The canonical view associated with the lowest score gives the winning pose and class label. The complexity of step-2 is just  $O(N_lN_s)$ , where  $N_l$  is the size of the short list and  $N_s$  is the number of interpolating steps (typically, 5 – 20).

## 8.7 Experiments and Results

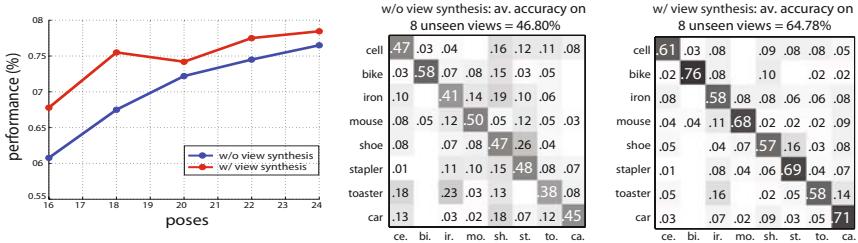
In this section we show that the multi-view model introduced so far is able to successfully detect object categories and estimate objects pose. Furthermore, we demonstrate that the view synthesis machinery does improve detection and pose estimation accuracy when compared to the model that does not take advantage of the synthesis abilities. We collect our results in three set of experiments as described below.

### 8.7.1 Experiment I: Comparison with Thomas et al. [63]

We first conduct experiments on two known 3D object class datasets: the motorbikes and sport shoes used by Thomas et al. [63], provided by PASCAL Visual Object Classes (VOC) Challenge [15]. For fair comparison, we use the same testing images in both these classes as in [63]. Specifically, 179 images from the ‘motorbikes-test2’ set and 101 images from the sport shoes testing set are used. The models are learnt by using the provided image set of 16 motorbike instances and 16 shoe instances (each instance has 12-16 poses). We evaluate the categorization and localization performance by using precision-recall curves, under exactly the same conditions as stated by [63]. Figure 8.11 illustrates our results. In both the motorbike and shoe classes, the proposed algorithm significantly outperforms [63].



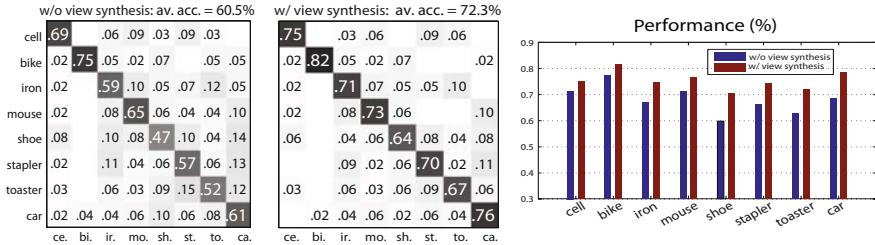
**Fig. 8.11** Localization experiment compared with [63]. The precision-recall curves are generated under the PASCAL VOC protocol. Example detections are shown for both the motor-bike and shoe datasets.



**Fig. 8.12** Left: Performances of the model with (red) and without (blue) view synthesis as a function of the number of views used in training. Note that the performances shown here are testing performances, obtained by an average over all 24 testing poses. Middle: Confusion table results obtained by the model without view synthesis for 8 object classes on a sample of 8 unseen views only (dataset [54]). Right: Confusion table results obtained by the model with view synthesis under the same conditions.

### 8.7.2 Experiment II: Detection and Pose Estimation Results on the Dataset in [54]

Next, we compare the performances of multi-view model algorithm with and without view-synthesis capabilities. The comparison is performed on the dataset presented in [54]. This dataset comprises images of 8 different object categories (car, stapler, iron, shoe, monitor, computer mouse, head, bicycle, toaster and cellphone), each containing 10 different instances. Each of these are photographed under a range of poses, described by a pair of azimuth and zenith angles (i.e., the angular coordinates of the observer on the viewing sphere, Figure 8.2) and distance (or scale). The total number of angular poses in this dataset is 24: 8 azimuth angles and 3 zenith angles. Each pose coordinate is kept identical across instances and categories. Thus,

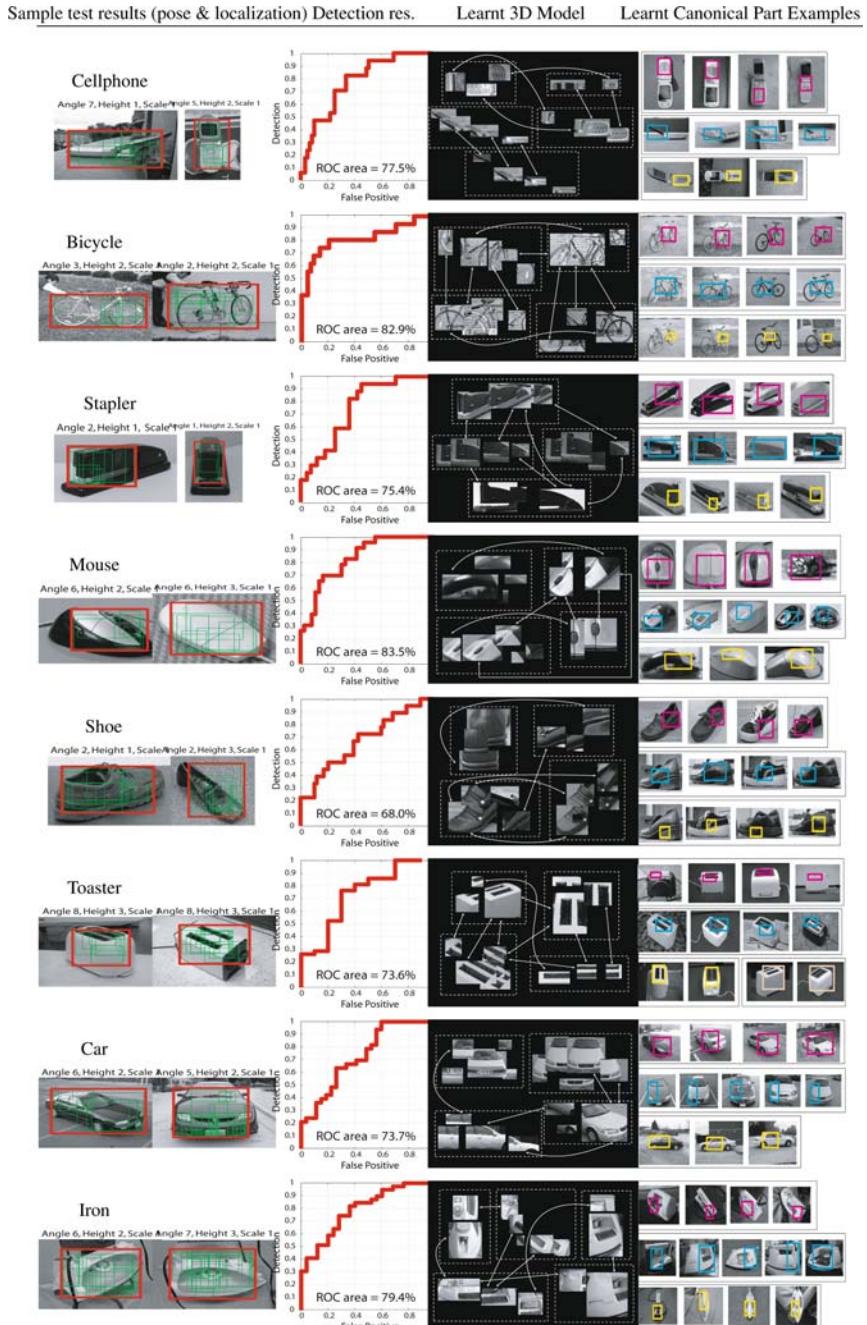


**Fig. 8.13** Left: Confusion table results obtained by the model without view synthesis for 8 object classes (dataset [55]). Middle: Confusion table results obtained by the model with view synthesis under the same conditions. Right: Performance improvement achieved by the model with view synthesis for each category.

the number and type of poses in the test set are the same as in the training set. To learn each category, we randomly select 7 object instances to build the model, and 3 novel object instances. The farthermost scale is not considered in the current results. Figure 8.14 is a summary of learnt models for 8 object categories. The 3<sup>rd</sup> column of Figure 8.14 visualizes the learnt model of each object category. We show in this panel a single object instance from the training images. Each dashed box indicate a particular view of the object instance. A subset of the learnt canonical parts is presented for each view. Across from different views, the canonical parts relationships are denoted by the arrows. Note that for clarity, we only visualize a small number of canonical parts as well as their  $\mathcal{H}$ . To illustrate the appearance variability, we show in the 4<sup>th</sup> column different examples of a given canonical part. For each object model, 3 or 4 canonical parts are shown, indicated by the boxes. For each canonical part (i.e., within each box), we show a number of examples that belong to the same part. Note that these parts not only share a similar appearance, but also similar locations with respect to the object. The 1<sup>st</sup> column of Figure 8.14 presents two correctly identified sample testing images. The red bounding box on each image indicates the best combination of canonical parts (i.e., that of the smallest error function), whereas the thin green boxes inside the red box correspond to the canonical parts of detected on the object. Using the pose estimation scheme, we are able to predict which pose this particular instance of the model comes from. Finally we present the binary detection result in ROC curves in the 2<sup>nd</sup> column.

To assess the ability of the view-synthesis algorithm to improve detection and pose estimation in presence of views that have not been seen in training, we tested the algorithm using a reduced set of poses in training. The reduced set is obtained by randomly removing poses from the original training set. This was done by making sure that no more than one view is removed from any quadruplet of adjacent poses in the viewing sphere<sup>1</sup>. The number of poses used in testing is kept constant (to be more specific, all 24 views are used in this case). This means some of the views in

<sup>1</sup> We have found experimentally that this condition is required to guarantee there are sufficient views for successfully constructing the linkage structure for each class.



**Fig. 8.14** Summary of the learnt 3D object category models, sample test images and binary detection results (ROC). Details of the figure is explained in Section 8.7.2



**Fig. 8.15** Estimated pose for each object that was correctly classified by the algorithm. Each row shows two test examples (the colored images in column 3 and column 6) from the same object category. For each test image, we report the estimated location of the object (red bounding box) and the estimated view-synthesis parameter  $s$ .  $s$  gives an estimate of the pose as it describes the interpolating factor between the two closest model (canonical) views selected by the recognition algorithm. For visualization purposes we illustrate these model views by showing the corresponding training images (columns 1-2 and 4-5). Images belong to the dataset in [55].

testing have not been presented during training. Figure 8.12 illustrates the performances of the two models (with and without view-synthesis) as a function of the number of views used in training. The plots shows that method which uses the two-step algorithm systematically outperforms the one that does only uses the first step. However, notice that the added accuracy becomes negligible as the number of views in training approaches 24. In other words, when no views are missing in training, the performance of two methods become similar. For a baseline comparison with a pure bag-of-world model the reader can refer to [54]. Figure 8.12(middle, right) compare the confusion table results obtained by the models with and without view-synthesis for 8 object classes on a sample of 8 unseen views only.

### 8.7.3 Experiment III: Detection and Pose Estimation Results on the Dataset in [55]

In this experiment we test the algorithm on a more challenging dataset [55]. While in [54] view points in testing and training are very similar, the dataset in [55] comprises objects portrayed under generic uncontrolled view points. Specifically, in [55] 7 (out of 8) classes of images (*cellphone, bike, iron, shoe, stapler, mouse, toaster*) are collected from the Internet (mostly Google and Flickr) by using an automatic image crawler. The initial images are then filtered to remove outliers by a paid undergraduate with no knowledge of the work so as to obtain a set of 60 images for each category. The 8<sup>th</sup> class (i.e., *car*) is from the LabelMe dataset [53]. A sample of the dataset is available at [55]. As in the previous experiment, we compare the performances of the algorithm with or without view-synthesis. Results by both models are reported in Figure 8.13. Again, the method that uses the two-step algorithm achieves better overall results. Figure 8.13 (right panel) shows the performance comparison broken down by each category. Notice that for some categories such as cellphone or bikes, the increment is less significant. All the experiments presented in this section use the 2-view synthesis scheme. The 3-view scheme, along with the introduction of a more sophisticated probabilistic model, has been recently employed in [61, 60]. Figure 8.15 illustrates a range of pose estimation results on the new dataset. See Figure 8.15 caption for details.

## 8.8 Conclusion

Recognizing objects in 3D space is an important problem in computer vision. Many works recently have been devoted to this problem. But beyond the possibility of semantic labeling of objects seen under specific views, it is often crucial to recognize the pose of the objects in the 3D space, along with their categorical identity. In this Chapter we have introduced a new recognition paradigm for tackling these challenging problems which consists of linking together diagnostic parts or features of the object from different viewing points. We focused on recent work by [54, 55] and presented the details of the multi-view part-based model in [54, 55], relevant learning and recognition algorithms, as well as practical implementation details. We

have shown that such a model can be learnt via minimal supervision and used to detect objects under arbitrary and/or unseen poses by means of a two-step algorithm. Experimental validation aimed at demonstrating the ability of the algorithm to recognize objects and estimate their pose have produced promising results. A number of open issues remain. The presented algorithms still require large number of views in training in order to generalize. More analysis needs to be done to make this as minimal as possible. Further research is also needed to explore to what degree the inherent *nuisances* in category-level recognition (lighting variability, occlusions and background clutter) affect the view synthesis formulation. Finally, new solutions are required for incorporating the ability to model non-rigid objects.

## References

1. The princeton shape benchmark. In: Proceedings of the Shape Modeling International, pp. 167–178 (2004)
2. Arie-Nachimson, M., Basri, R.: Constructing implicit 3d shape models for pose estimation. In: Proceedings of the International Conference on Computer Vision (2009)
3. Ballard, D.H.: Generalizing the hough transform to detect arbitrary shapes. Pattern Recognition 13(2), 111–122 (1981)
4. Bart, E., Byvatov, E., Ullman, S.: View-invariant recognition using corresponding object fragments. In: Pajdla, T., Matas, J.(G.) (eds.) ECCV 2004. LNCS, vol. 3022, pp. 152–165. Springer, Heidelberg (2004)
5. Bowyer, K., Dyer, R.: Aspect graphs: An introduction and survey of recent results. International Journal of Imaging Systems and Technology 2(4), 315–328 (1990)
6. Brown, M., Lowe, D.G.: Unsupervised 3d object recognition and reconstruction in un-ordered datasets. In: Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling, pp. 56–63 (2005)
7. Burl, M.C., Weber, M., Perona, P.: A probabilistic approach to object recognition using local photometry and global geometry. In: Burkhardt, H., Neumann, B. (eds.) ECCV 1998. LNCS, vol. 1407, p. 628. Springer, Heidelberg (1998)
8. Chen, S., Williams, L.: View interpolation for image synthesis. Computer Graphics 27, 279–288 (1993)
9. Chiu, H.P., Kaelbling, L.P., Lozano-Perez, T.: Virtual training for multi-view object class recognition. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2007)
10. Cyr, C., Kimia, B.: A similarity-based aspect-graph approach to 3D object recognition. International Journal of Computer Vision 57(1), 5–22 (2004)
11. Dance, C., Willamowski, J., Fan, L., Bray, C., Csurka, G.: Visual categorization with bags of keypoints. In: Proceedigs of the ECCV International Workshop on Statistical Learning in Computer Vision (2004)
12. Dickinson, S.J., Pentland, A.P., Rosenfeld, A.: 3-d shape recovery using distributed aspect matching. IEEE Transaction on Pattern Analisys and Machine Intelligence 14(2), 174–198 (1992)
13. Eggert, D., Bowyer, K.: Computing the perspective projection aspect graph of solids of revolution. IEEE Transaction on Pattern Analisys and Machine Intelligence 15(2), 109–128 (1993)
14. Eggert, D., Bowyer, K., Dyer, C., Christensen, H., Goldgof, D.: The scale space aspect graph. IEEE Transactions on Pattern Analysis and Machine Intelligence 15(11), 1114–1130 (1993)

15. Everingham, M., et al.: The 2005 pascal visual object class challenge. In: Proceedings of the 1st PASCAL Challenges Workshop (to appear)
16. Farhadi, A., Tabrizi, J., Endres, I., Forsyth, D.: A latent model of discriminative aspect. In: Proceedings of the International Conference on Computer Vision (2009)
17. Fei-Fei, L., Fergus, R., Torralba, A.: Recognizing and learning object categories. CVPR Short Course (2007)
18. Felzenszwalb, P., Huttenlocher, D.: Pictorial structures for object recognition. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 2066–2073 (2000)
19. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 264–271 (2003)
20. Ferrari, V., Tuytelaars, T., Van Gool, L.: Simultaneous object recognition and segmentation from single or multiple model views. International Journal of Computer Vision (2006)
21. Fischler, M., Bolles, R.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Comm. of the ACM 24, 381–395 (1981)
22. Frome, A., Huber, D., Kolluri, R., Bulow, T., Malik, J.: Recognizing objects in range data using regional point descriptors. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3023, pp. 224–237. Springer, Heidelberg (2004)
23. Fulkerson, B., Vedaldi, A., Soatto, S.: Class Segmentation and Object Localization with Superpixel Neighborhoods. In: Proceedings of the International Conference on Computer Vision (2009)
24. Grimson, W., Lozano-Perez, T.: Recognition and localization of overlapping parts in two and three dimensions. In: Proceedings of the International Conference on Robotics and Automation, pp. 61–66 (1985)
25. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2004)
26. Hetzel, G., Leibe, B., Levi, P., Schiele, B.: 3d object recognition from range images using local feature histograms. IEEE Transactions on Pattern Analysis and Machine Intelligence 2 (2001)
27. Hoeim, D., Rother, C., Winn, J.: 3d layout crf for multi-view object class recognition and segmentation. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
28. Johnson, A., Hebert, M.: Using spin images for efficient object recognition in cluttered 3d scenes. IEEE Transactions on Pattern Analysis and Machine Intelligence 5 (1999)
29. Kadir, T., Brady, M.: Scale, saliency and image description. International Journal of Computer Vision 45(2), 83–105 (2001)
30. Kazhdan, M., Funkhouser, T., Rusinkiewicz, S.: Rotation invariant spherical harmonic representation of 3d shape descriptors. In: Proceedings of the Symposium on Geometry Processing (2003)
31. Koenderink, J., van Doorn, A.: The singularities of the visual mappings. Biological Cybernetics 24(1), 51–59 (1976)
32. Koenderink, J.J., van Doorn, A.J.: The internal representation of solid shape with respect to vision. Biological cybernetics 32(4), 211–216 (1979)
33. Kushal, A., Schmid, C., Ponce, J.: Flexible object models for category-level 3d object recognition. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
34. Lazebnik, S., Schmid, C., Ponce, J.: Semi-local affine parts for object recognition. In: Proceedings of the British Machine Vision Conference, vol. 2, pp. 959–968 (2004)

35. Leibe, B., Schiele, B.: Scale Invariant Object Categorization Using a Scale-Adaptive Mean-Shift Search. In: Rasmussen, C.E., Bültmann, H.H., Schölkopf, B., Giese, M.A. (eds.) DAGM 2004. LNCS, vol. 3175, pp. 145–153. Springer, Heidelberg (2004)
36. Li, X., Guskov, I., Barhak, J.: Feature-based alignment of range scan data to cad model. *International Journal of Shape Modeling* 13, 1–23 (2007)
37. Liebelt, J., Schmid, C., Schertler, K.: Viewpoint-independent object class detection using 3d feature maps. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2008)
38. Lowe, D.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision, pp. 1150–1157 (1999)
39. Lowe, D.G.: Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence* 31, 355–395 (1987)
40. Lowe, D.G.: Local feature view clustering for 3d object recognition. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2001)
41. Marr, D.: Vision: A computational investigation into the human representation and processing of visual information. Freeman, New York (1982)
42. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: Proceedings of the British Machine Vision Conference, pp. 384–393 (2002)
43. Mei, L., Sun, M., Carter, K., Hero, A., Savarese, S.: Object pose classification from short video sequences. In: Proceedings of the British Machine Vision Conference (2009)
44. Mikolajczyk, K., Schmid, C.: An affine invariant interest point detector. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2350, pp. 128–142. Springer, Heidelberg (2002)
45. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. *International Journal of Computer Vision* 60(1), 63–86 (2004)
46. Murase, H., Nayar, S.K.: Learning by a generation approach to appearance-based object recognition. In: Proceedings of the International Conference on Pattern Recognition (1996)
47. Nayar, S.K., Nene, S.A., Murase, H.: Real-time 100 object recognition system. In: Proceedings of the International Conference on Robotics and Automation, pp. 2321–2325 (1996)
48. Ng, J., Gong, S.: Multi-view face detection and pose estimation using a composite support vector machine across the view sphere. In: Proceedings of the International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (1999)
49. Ozuysal, M., Lepetit, V., Fua, P.: Pose estimation for category specific multiview object localization. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2009)
50. Rothganger, F., Lazebnik, S., Schmid, C., Ponce, J.: 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision* 66(3), 231–259 (2006)
51. Rothwell, C.A., Zisserman, A., Forsyth, D.A., Mundy, J.L., Joseph, L.: Canonical frames for planar object recognition. In: Sandini, G. (ed.) ECCV 1992. LNCS, vol. 588. Springer, Heidelberg (1992)
52. Ruiz-Correa, S., Shapiro, L., Meila, M.: A new signature-based method for efficient 3-d object recognition. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2001)
53. Russell, B., Torralba, A., Murphy, K., Freeman, W.: Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision* (in press)

54. Savarese, S., Fei-Fei, L.: 3D generic object categorization, localization and pose estimation. In: Proceedings of the International Conference on Computer Vision, pp. 1–8 (2007)
55. Savarese, S., Fei-Fei, L.: View synthesis for recognizing unseen poses of object classes. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) *ECCV 2008, Part III*. LNCS, vol. 5304, pp. 602–615. Springer, Heidelberg (2008)
56. Schneiderman, H., Kanade, T.: A statistical approach to 3D object detection applied to faces and cars. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 746–751 (2000)
57. Seitz, S., Dyer, C.: View morphing. In: Proceedings of the ACM SIGGRAPH, pp. 21–30 (1996)
58. Shimshoni, I., Ponce, J.: Finite-resolution aspect graphs of polyhedral objects. *IEEE Transaction on Pattern Analysis Machine Intelligence* 19(4), 315–327 (1997)
59. Stewman, J., Bowyer, K.: Learning graph matching. In: Proceedings of the International Conference on Computer Vision, pp. 494–500 (1988)
60. Su, H., Sun, M., Fei-Fei, L., Savarese, S.: Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In: Proceedings of International Conference on Computer Vision (2009)
61. Sun, M., Su, H., Savarese, S., Fei-Fei, L.: A multi-view probabilistic model for 3d object classes. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2009)
62. Tanger, J.W.H., Veltkamp, R.C.: A survey of content based 3d shape retrieval methods. In: Proceedings of Shape Modeling Applications, pp. 145–156 (2004)
63. Thomas, A., Ferrari, V., Leibe, B., Tuytelaars, T., Schiele, B., Van Gool, L.: Towards multi-view object class detection. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 1589–1596 (2006)
64. Torralba, A., Murphy, K., Freeman, W.: Sharing features: efficient boosting procedures for multiclass object detection. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2004)
65. Ullman, S., Basri, R.: Recognition by linear combination of models. Technical Report, Cambridge, MA, USA (1989)
66. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 511–518 (2001)
67. Weber, M., Einhäuser, W., Welling, M., Perona, P.: Viewpoint-invariant learning and detection of human heads. In: Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (2000)
68. Weber, M., Welling, M., Perona, P.: Unsupervised learning of models for recognition. In: Vernon, D. (ed.) *ECCV 2000*. LNCS, vol. 1842, pp. 101–108. Springer, Heidelberg (2000)
69. Xiao, J., Chen, J., Yeung, D.Y., Quan, L.: Structuring visual words in 3d for arbitrary-view object localization. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) *ECCV 2008, Part III*. LNCS, vol. 5304, pp. 725–737. Springer, Heidelberg (2008)
70. Yan, P., Khan, D., Shah, M.: 3d model based object class detection in an arbitrary view. In: Proceedings of the International Conference on Computer Vision (2007)
71. Yan Li Leon Gu, T.K.: A robust shape model for multi-view car alignment. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2009)
72. Zhang, Z.: Floatboost learning and statistical face detection. *IEEE Transaction on Pattern Analysis Machine Intelligence* 26(9), 1112–1123 (2004)

# Chapter 9

## A Vision-Based Remote Control

Björn Stenger, Thomas Woodley, and Roberto Cipolla

**Abstract.** This Chapter presents a vision-based system for touch-free interaction with a display at a distance. A single camera is fixed on top of the screen and is pointing towards the user. An attention mechanism allows the user to start the interaction and control a screen pointer by moving their hand in a fist pose directed at the camera. On-screen items can be chosen by a selection mechanism. Current sample applications include browsing video collections as well as viewing a gallery of 3D objects, which the user can rotate with their hand motion. We have included an up-to-date review of hand tracking methods, and comment on the merits and shortcomings of previous approaches. The proposed tracker uses multiple cues, appearance, color, and motion, for robustness. As the space of possible observation models is generally too large for exhaustive online search, we select models that are suitable for the particular tracking task at hand. During a training stage, various off-the-shelf trackers are evaluated. From this data different methods of fusing them online are investigated, including parallel and cascaded tracker evaluation. For the case of fist tracking, combining a small number of observers in a cascade results in an efficient algorithm that is used in our gesture interface. The system has been on public display at conferences where over a hundred users have engaged with it.

### 9.1 Introduction

This Chapter presents a vision-based gesture interface using a single camera on top of a display, allowing touch-free input at a distance. Figure 9.1 shows photos from

---

Björn Stenger,  
Computer Vision Group, Toshiba Research Europe, Cambridge, UK  
e-mail: [bjorn@cantab.net](mailto:bjorn@cantab.net)

Thomas Woodley  
Department of Engineering, University of Cambridge, UK  
e-mail: [ew32@cam.ac.uk](mailto:ew32@cam.ac.uk)

Roberto Cipolla  
Department of Engineering, University of Cambridge, UK  
e-mail: [cipolla@eng.cam.ac.uk](mailto:cipolla@eng.cam.ac.uk)



**Fig. 9.1** Showcase of the proposed gesture interface. Left: a single camera on top of the display is directed towards the user, who is able to control a screen cursor with his fist. Videos can be selected by hovering over a button. Right: The playback of a video can be stopped with an open hand gesture.



**Fig. 9.2** Appearance variation of hand regions. Shown are cropped hand regions from test sequences. Motion blur, changing pose and other skin colored objects make tracking challenging.

a showcase presented during the *Internationale Funk Ausstellung IFA 2008* exhibition in Berlin. Such a gesture system may have several uses in practice: to remotely control a TV or other appliances, or browsing public information terminals in museums or shop windows. There are many factors that make hand tracking from a single view difficult in practice. Hands can exhibit a wide range of appearances, for example due to changes in pose and in scene lighting. Furthermore, the variability of shapes, poses, and color between different people is high. At a standard frame rates of 30 frames per second there may also be significant motion blur. Figure 9.2 illustrates this variability, showing examples of cropped hand regions, taken from image sequences of four different people. The system should also use minimal computational resources because any lag in interactive applications is very noticeable.

In addition to fast and robust tracking, a method for automatic initialization is required to find the hand at the beginning of the interaction and after tracking failure. Loss of track occurs regularly, for example every time the hand is outside the camera's view. The proposed system thus integrates an off-line trained detector to initialize and also to update the tracker in order to avoid drift. Building a general, robust hand detector is still a challenging problem, and we restrict ourselves

to detecting a particular pose, in this case a fist pose. When viewed from the front, the visual appearance of a fist is characteristic and a robust detector has been trained for it.

In the following Section we provide an overview of prior work on hand tracking and object tracking in general. Section 9.3 explains the design of the tracking algorithm and presents experimental results on test data. The full system and its components are described in Section 9.4.

## 9.2 Prior Work

A large number of vision-based gesture interfaces have been proposed in the scientific literature and some systems have already been commercialized. This Section provides an overview of hand tracking methods in image sequences. It also summarizes developments in the area of general object tracking as some of these methods are used in Section 9.3.

### 9.2.1 Hand Tracking for Human Computer Interfaces

Reviews on hand tracking have been published by Pavlović *et al.* [62], Wu and Huang [86,88]. These papers contain a good taxonomy of early work on hand tracking and gesture interfaces. More recently, Erol *et al.* [24] published a review of full 3D hand tracking. Generally, there are a number of factors to consider when designing or describing a hand tracking system.

1. **The number of state parameters.** Methods differ by the number of parameters they estimate. This may range from the case where only the 2D location of a hand in an image is obtained to the case where the full articulated pose in 3D is estimated. In some cases a dynamic model is used, whose parameters are included in the estimation process.
2. **The estimation method and features used.** At the heart of each system is the algorithm which estimates the state parameters from the observed image data. Some methods employ an explicit geometric model and use a model-fitting approach, while others take the approach of learning from training data. Hybrid approaches exist too, which generate training data from a geometric model. Methods also differ in the types of features they use, which can be based on color, shape, or motion of the hand.
3. **The set-up and capture system.** There exist a wide range of set-ups, differing in the number and position of cameras, for example. Systems that use two or more cameras may compute a depth map or a visual hull as the input to the recognition system. Furthermore, active systems such as structured light or time-of-flight systems are becoming more common and allow depth estimation that is often more robust than passive two-view stereo. Other important parameters are the camera's resolution, light sensitivity and frame-rate. Clearly the camera position also makes a difference: whether it is facing top-down towards a desktop, whether

it is facing towards a user in front of an uncontrolled background, or whether it is mounted on a mobile robot platform.

These factors should be kept in mind whenever analyzing a gesture interface system, as the underlying assumptions differ in each case. In the following we introduce some hand tracking systems that have been prominent in the literature.

One of the first systems for markerless hand tracking was that of Cipolla and Hollinghurst [15, 16], who used a B-spline active contour as a 2D shape model to track a pointing hand from two uncalibrated views. In each view an affine transformation of the contour was estimated, and using a ground plane constraint the indicated target position could be found. The system required a simple background, but it could operate in real-time. Freeman and Weissman [26] introduced a single-camera system for television remote control by tracking an open hand. It used an image subwindow as a template for both detection and tracking. Matching was performed using local edge orientation in order to be robust to some illumination changes. In their *EigenTracking* work, Black and Jepson [10] proposed an eigenspace representation of a set of hand images. Using a coarse-to-fine matching strategy, both the affine transformation as well as the closest of four gestures were estimated. Isard and Blake [40] modeled the hand shape with a B-spline. The tracker combined color blob tracking with contour tracking in a particle filter framework. Later, MacCormick and Isard [52] presented a drawing system based on a this tracker together with independent sampling of the finger parameters. Tosas [76] recently presented a similar color-based contour tracker in a number of demo applications, such as a ‘virtual turntable’. Wu and Huang [87] applied a learning based method, combining labeled and unlabeled data with the EM algorithm and using neural network classification to distinguish 14 hand postures in different views. The classification rate was 92% using features obtained by Principal Component Analysis of the hand images. Triesch and von der Malsburg [78] built a hand gesture interface for robot control. Elastic 2D graph matching was employed to match templates of twelve different hand gestures to an image. Combined Gabor and color features made the system relatively stable to clutter, achieving a recognition rate of 93% in front of simple background and 86% for cluttered backgrounds. Bretzner *et al.* [11] used multi-scale blob detection of color features in order to detect an open hand pose with possibly some of the fingers extended, corresponding to different input commands. A simple 2D shape model was used for tracking with a particle filter. The method required a skin color prior, which was obtained by manually labeling 30 frames. The system tracked at 10 fps and was also demonstrated in a TV remote control application. Lockton and Fitzgibbon [50] built a real-time system that could recognize 46 different hand poses, including finger spellings in American Sign Language. Extracting hand silhouettes in each frame using color, their method was based on efficient template matching. Accurate registration was facilitated by a wrist band and led to a very high recognition rate. Krahnstoever *et al.* [47] presented a multi-modal vision and speech interface to interact with a large display. The hand tracking component was based on finding location hypotheses in each frame and matching them over a time-window with the Viterbi algorithm. A spatial prior was used to associate blobs to hand and face. An interface based on tracking *multiple*

skin colored regions was proposed by Argyros and Lourakis in [1]. The skin color model was obtained by manually labeling skin regions, but the color model was adapted during tracking. The tracker was used in a stereo-system in order to realize a 3D mouse application [2]. Extensions of cascaded detection using AdaBoost were proposed in [46, 60]. Whereas Kölisch and Turk [46] showed robust detection of specific hand poses, Ong and Bowden [60] trained a classifier hierarchy for multi-pose detection. Kölisch and Turk [45] further presented a multi-cue tracker that combined color and a number of local features under ‘flocking’ constraints. The color model was automatically initialized from hand detection. The system by Robertson *et al.* [67] used a trained detector followed by optical flow tracking and was employed in a ‘virtual mouse’ application. Ike *et al.* [37] presented a real-time system for gesture control that could detect three different hand poses independently in each frame. Due to the high computation requirement it was implemented on a multi-core processor.

To summarize, a large number of hand trackers for gesture recognition have been proposed in the literature. We have re-implemented some of these for evaluation.

One class of applications, which this Chapter does not discuss in detail are *virtual desktop* applications [42, 58, 70, 83]. Originally, in most of these systems the camera and possibly a projector are mounted above, facing down towards the tabletop and a user is able to interact with real and projected virtual objects. Similarly, by placing the camera below a transparent tabletop a multi-touch interface can efficiently be implemented, such as in *Microsoft Surface* [53].

In addition to gesture-based control, a further target application of hand tracking is automatic sign language recognition. The pioneering recognition system by Starner *et al.* [69] modeled a hand by a skin-colored ellipse that was tracked in image sequences. A hidden Markov model was then used to recognize a 40 word vocabulary based on the shape and motion trajectory, obtaining a recognition rate of 98%. Much progress has been made recently, see [61] for a general survey, and [12, 20] for state-of-the-art results.

For some applications, such as motion capture for animation or biomechanical analysis, it is desirable to fully capture the hand motion in 3D. Currently these methods use either colored gloves, markers on the hand, or data gloves with built-in sensors. However, there has been progress in markerless 3D hand tracking [3, 28, 35, 66, 68, 72]. A common approach to full 3D hand tracking is to use a geometric hand model. The model is usually created manually, but can also be obtained by reconstruction methods. Models that have been used for tracking are based on planar patches [89, 90], deformable polygon meshes [35] or generalized cylinders [21, 66]. The underlying kinematic structure is based on the biomechanics of the hand. Each finger can be modeled as a kinematic chain with four degrees of freedom (DOF) attached to the palm, and the thumb may be modeled similarly with either four or five DOF. Together with rigid body motion of the hand, there are 27 DOF to be estimated. Working in such a high dimensional space is particularly difficult because feature points that can be tracked reliably are sparse: the texture on hands is typically weak and self-occlusion occurs frequently. However, the anatomical design places certain constraints on the hand motion. These constraints have

been exploited in different ways to reduce the search space. One type of constraint are the joint angle limits, defining the range of motion. Another very significant constraint is the correlation of joint movement. Angles of the same finger, as well as of different fingers, are not completely independent. These constraints can be enforced in different ways, either by parameter coupling or by working in a space of reduced dimension, e.g., one found by PCA [35,89]. To summarize, while headway has been made on the problem of full 3D tracking, the task remains challenging. It has been shown to work in principle, but is often constrained to slow hand motion or controlled scenes. The problem is generally ill-posed using a single camera, because in some poses finger motion can be unobservable due to self-occlusion. The use of additional hardware, such as multiple cameras [13,32], colored gloves [82] or depth-cameras [34], can therefore help significantly.

### 9.2.2 Commercial Gesture Interface Systems

A number of commercial systems for gesture recognition have already been made available and more are likely to follow.

Game console makers have shown interest in gesture recognition, while proposing different solutions. While Sony has been working with a passive vision system [25,64], Nintendo is using a wireless controller with accelerometer and optical sensing technology [56] and Microsoft has shown technology based on active illumination [65]. One of the first widely known vision systems was the *EyeToy* camera in 2003 [25] that could be connected to the *Play Station 2* game console and its successor in 2007, the *PlayStation Eye* for the PS3 [64]. Here the camera is placed on top of the screen, facing the user who can interact with menus and games using gestures. In gaming, it is clearly important to avoid any lag that can interfere with game play. Another key requirement is to handle dark or changing lighting conditions in living rooms. Until now, the algorithms used for gesture recognition used in *EyeToy* games have been fairly basic yet sufficiently robust. Two examples are optical flow estimation and frame differencing in order to find regions of object motion.

Oblong Industries, co-founded by John Underkoffler, commercialized *g-speak*, an interface using gloves and markers [57]. Underkoffler was also a scientific adviser for the 2002 science fiction movie *Minority Report* which featured a similar gesture interface for video navigation.

In order to overcome many of the robustness issues, active systems such as time-of-flight cameras, have been used for gesture recognition, e.g., [14]. These systems directly output a depth map, considerably simplifying feature extraction and segmentation. Companies such as *GestureTek* have commercialized a number of gesture recognition systems using time-of-flight cameras, for example systems for public exhibitions [27].

In 2008 the *Toshiba Qosmio* laptop first shipped with hand gesture control software that worked with the built-in webcam [77]. The gesture system works by integrating global hand detection [38,54] with local tracking. Users are able to control a

screen pointer with their hand and select objects by moving their thumb. The work presented this Chapter mainly builds on this system.

### 9.2.3 Visual Tracking of a Single Object

In the following we take a more general view and consider the task of robustly tracking a single object in image sequences. This has been studied extensively in the computer vision literature [9, 19, 33, 36, 39, 75, 79, 85]. The task can be stated as follows: Given an initial state of the object in the first frame, estimate the state of the object in the subsequent frame, then do this sequentially on each frame of the sequence. The state of the object contains the parameters of the geometric transformation that we are interested in. This can be, for example, the target's  $x$ - $y$ -position in the image or the full three-dimensional pose. Some methods also estimate the shape or articulation parameters of a target object. Tracking methods also differ in the image cues they employ to measure the similarity from frame to frame. These can be raw or normalized image pixels, edge contours, color histograms, outputs of oriented filters, or any other features computed from the object region. In order to successfully track the object in the next frame, the underlying assumption is that the features are still sufficiently characteristic for the object (*generative* approaches), or make it appear different from the background (*discriminative* approaches). The chance of finding discriminative features is clearly increased when combining multiple cues. Numerous papers on multi-cue tracking have demonstrated the concept of different cues complementing each other and overcoming the failure cases of individual cues [6, 9, 23, 31, 48, 55]. A typical example is a hand being tracked while it moves in front of the face. The hand may still be tracked based on shape while color features become less reliable. The most common approach to multi-cue tracking is to evaluate several observers in parallel and subsequently combine their output, by either switching between them [6] or by probabilistically merging them [23, 48, 55, 63]. A key issue when merging tracking results is how to obtain a good confidence measure for each cue. This is a tricky question since the performance of one cue may only be assessed by using a different cue or different representation of the target object. One answer is the discriminability between foreground object and background region. This is the basis of recent work on *discriminative* tracking [5, 17, 29], where tracking is formulated as a classification task. Collins *et al.* proposed a method for online feature selection which selects the most discriminative features from a pool of color-based features [18]. The ability to discriminate was evaluated as either the two-class variance ratio or the difference of the top two likelihood peaks. Avidan introduced *ensemble tracking*, where multiple (3-5) weak classifiers were combined via AdaBoost [5]. At each frame a new weak classifier was learned and the ensemble was updated by replacing the least reliable classifiers in each time step. Grabner and Bischof applied online boosting to feature selection [29]. Features from a larger pool of 250 weak classifiers were evaluated and a set of 50 selectors chose those that were combined into a strong classifier.

In practice, an issue with online adaptation is the *adaptability vs. drift* trade-off: Allowing the tracker to adapt to rapid changes of the object's appearance bears the risk of incorrectly adapting to the background. Ideally an object model is available that includes all possible variations. Such a model could then be used as an 'anchor' for the tracker. Obtaining such a model is challenging and different representations have been used, including the color distribution [6], a representation learned from a short initial sequence [30] or an off-line trained detector [4,45,84]. Detectors have been included into tracking systems, for example, by running them in tandem [45, 84] or by closely integrating them with the tracker's estimation procedure [4,49,59]. Indeed, a viable tracking solution is to use a detect-and-connect strategy, shown for example in [44]. In many cases this approach is not yet sufficiently fast for real-time tracking and the detectors lack sufficient flexibility, but this is a promising avenue for future research.

Existing algorithms that integrate multiple cues choose their component observers in a heuristic manner beforehand [6,9,49]. In the following we are proposing a method to make this choice in a more principled way.

## 9.3 Tracking with Multiple Observers

We now address the question of how to design a tracker using multiple observation models. The observation models are components from different stand-alone tracking algorithms such as single template matching, optical flow and online classification. The idea is to learn which of these are suitable components and how they should be arranged for efficient evaluation. We collect a set of training sequences and ground-truth label them by hand. On these sequences we evaluate error distributions for different observers. The ground truth labels allow us to evaluate combinations of observers on a test set. The term 'observer' in this Chapter refers to an observation model, and can be seen as a component of a tracker, the other component being the dynamical model. However, in the evaluation we only rely on the observations within a search window around the previous estimate, thus both terms may be used interchangeably here. We consider the particular tracking scenario of tracking a fist using a static camera. However, the method is general and can be applied to other settings [73].

### 9.3.1 Observation Models

The goal is to find, for a given tracking scenario, the best observer or combination of observers. Our approach is to first evaluate each observer individually and from these values measure the performance of combinations of observers. The observers we consider are those used previously in tracking algorithms, see Table 9.1. They can be classified into four types: single template matching, motion consensus of local features [6,45,51], histogram-based region matching [19] and online classification [17,29,49]. Note that the individual observers are not restricted to using a single cue.

**Table 9.1** Observers in the evaluation. A diverse range of observers are tested in the experiments. They can be grouped into four types: single template matching, local feature matching, histogram-based region matching, and online classifiers. Between them they use a variety of cues, including image intensity, color and motion features. Some observers maintain a fixed representation while others are updated over time.

Method	Observation	Estimate	Confidence value
NCC	Normalized cross correlation	max correlation	correlation score
SAD	Sum of absolute differences	min distance	distance score
BOF	Block-based optical flow of $3 \times 3$ templates	mean motion	mean NCC score
KLT [51]	Kanade-Lucas-Tomasi sparse optical flow using 50 features	centroid of good features	fraction of good features
FF [45]	Flocks of features: Tracking 50 local features with high color probability and ‘flocking’ constraints	centroid of good features	fraction of good features
RT [6]	Randomized templates: NCC track of eight subwindows, with motion consensus and resampling	centroid of good features	fraction of good features
MS [19]	Mean shift: Color histogram-based mean shift tracking with background weighting	min histogram distance	histogram distance
C [71]	Color probability map, blob detection	scale space maximum	probability score
M [71]	Motion probability map, blob detection	scale space maximum	probability score
CM [47]	Color and motion probability map	scale space maximum	probability score
OBD [29]	Online boosted detector: Classifier boosted from pool of rectangle features updated online	max classifier output	classifier margin
LDA [49]	LDA classifier computed from five rectangle features in the previous frame (Observer 1 in [49])	max classifier output	classifier margin
BLDA [49]	Boosted LDA classifier using 50 LDA classifiers from a pool of 150, trained on the previous five frames (Observer 2 in [49])	max classifier output	classifier margin
OFS [17]	Online feature selection of 3 out of 49 color-based features based on fg/bg variance ratio	centroid of top features	mean variance ratio of selected features

**Single Template Matching.** The first two observers use normalized cross correlation (NCC), and sum of absolute differences (SAD), respectively. Given the subwindow of the most recent detection, the matching score in the search window is computed exhaustively. We also performed initial experiments using correlation of local orientations, used by Freeman and Weissman [26]. We found that the tracker works when the hand moves slowly, but edge features tend to be unstable when motion blur occurs.

**Local Feature Matching.** The first local feature method is block-based optical flow (BOF), where the object region is divided into a regular  $3 \times 3$  grid of subwindows which are matched independently in the next frame using NCC. The second method computes sparse optical flow on salient features, using the Kanade-Lucas-Tomasi (KLT) tracker with a set of 50 features [8, 45]. The third method is the ‘flocks of features’ tracker by Kölsch and Turk [46]. It combines motion cues with a learned

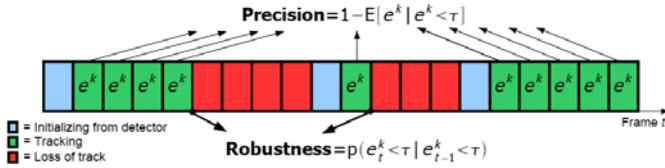
object color distribution. KLT features are used to compute motion which has to satisfy the following constraints constraints: (1) all features maintain a minimum distance from each other, and (2) no feature is further from the current median than a maximum distance. The name of the method comes from the similarity to the motion pattern of flocks of birds. A further constraint is that the features need to be located in regions of high object color probability. Features that do not satisfy all three constraints are pruned and replaced with new ones. Finally, we have re-implemented the randomized template method of [6]. A set of eight templates that are randomly located and sized within the object region is tracked using NCC.

**Region Matching.** The first method in this category is color based mean shift [19], where the best match is found by minimizing the distance of the color distribution to the target. We use an RGB histogram representation, where each color channel is divided into 16 bins. In addition we use the background color weighting scheme that was proposed as an extension in [19]. The next method is based on pixel-wise color probabilities as proposed by Jones and Rehg for skin color detection [43]. Color distributions of the object and the surrounding background region are obtained during initialization and the probability for each pixel belonging to the foreground is computed. We then run a box filter over this probability image that finds blobs, i.e., regions of high object probability surrounded by regions of low object probability [7]. This is quite similar to the hand tracker by Bretzner *et al.* [11] where scale-space extrema in color feature space are found. The third method uses the motion cue in a similar way. It computes the pixel-wise probability of motion in a region. The distributions for moving regions and background are obtained off-line from a hand labeled sequence of frame difference images. Note that in general this cue is typically present near the object's boundary, but not necessarily inside the object for homogeneous surfaces. The final method combines both color and motion cues. The function combines three terms as a weighted sum as in [47]. The functions are smoothed spatially by Gaussians with a variance depending on the size of the previously detected hand. The pixel-wise probability density function of observing a hand at image location  $\mathbf{y}$  is defined as

$$p(\text{hand}|\mathbf{y}) \propto w_c p(\mathbf{y}|\text{col}) + w_m p(\mathbf{y}|\text{mot}) + (1 - w_c - w_m) p(\mathbf{y}|\text{col}) p(\mathbf{y}|\text{mot}), \quad (9.1)$$

where  $w_c$  and  $w_m$  are weights that are determined through experiments on a validation set (in our case  $w_c = w_m = 0.1$ ).

**Online Classification.** The first method in this category is a re-implementation of online boosting as proposed by Grabner and Bischof [29]. A classifier is learned online by selecting a set of rectangle features (weak classifiers) from a pre-selected pool of 250 features. Assuming that the object location is known at frame  $t$ , the classifier is evaluated in the neighborhood of the previous location to create a confidence map. The new object location is moved to its maximum and, using the new labels for object and background, new training examples are sampled from the image to update the classifier. The second online method uses linear discriminant analysis (LDA), based on five rectangle features. The third method is Boosted LDA and



**Fig. 9.3** Observer evaluation. At each frame  $t$  of a test sequence an observer  $O^k$  outputs its position estimate  $\hat{\mathbf{x}}_t^k$  and confidence value  $c_t^k$ . The position error  $e_t^k$  relative to the ground truth is calculated during successful tracking (represented by green cells). Loss of track occurs when the error exceeds a threshold  $\tau$  (switch to red). Tracking is re-initialized from an off-line trained detector (blue). Precision and robustness metrics are calculated from the test results.

combines 50 classifiers from a pool of 150 using AdaBoost. These two methods have been proposed by Li *et al.* [49] who used these as observation models in a particle filter trained over different time periods. While the LDA classifier is trained only on the previous frame, the Boosted LDA classifier is trained on the previous five frames. Finally, we have implemented the online feature selection (OFS) scheme by Collins *et al.* [18], where discriminative color features are found using the variance ratio criterion.

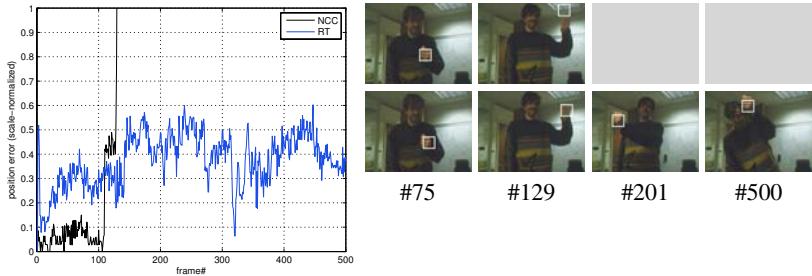
We train a detector to initialize and re-initialize each tracker. The detector was trained off-line using AdaBoost for feature selection [54, 81] from a dataset of approximately 5000 positive and negative examples. The version that we use [54] uses feature co-occurrence to increase the performance over the baseline method.

### 9.3.2 Evaluating Single Observers

The evaluation of observers proceeds as follows. Given an image sequence  $I_t, t = 1, \dots, T$ , at every time step  $t$  each observer  $O^k, k = 1, \dots, K$  computes an estimate of the target location  $\hat{\mathbf{x}}_t^k$  as well as the distance to the labeled ground truth location  $\mathbf{x}_t^{gt}$  as error  $e_t^k = d(\hat{\mathbf{x}}_t^k, \mathbf{x}_t^{gt})$ . The estimate  $\hat{\mathbf{x}}_t = (x, y, s)$  contains the center location  $x, y$  and scale estimate  $s$ . The error is computed as the the scale-normalized distance between the centers. Observers that do not estimate the scale  $s$ , obtain this value from the most recent detection and it remains constant during tracking. Every observer also outputs a confidence value  $c_t^k$  at each frame, which is computed depending on the type of observer, see the right column of Table 9.1. Loss of track occurs when the location error  $e_t^k$  is above a threshold value  $\tau < 1$ . In this case the tracker outputs  $\tau$  as error and is re-initialized at the next successful detection. The detections are pre-computed by running the off-line detector over all sequences. In summary, the measurements for the individual observers  $O_k$  on a training sequence are given by

$$\mathcal{Z}^k = \{\hat{\mathbf{x}}_t^k, e_t^k, c_t^k\}, \quad t = 1, \dots, T. \quad (9.2)$$

This allows the evaluation of single observers on the complete sequence, not just on the first successfully tracked segment. Loss of track can occur at any time during the



**Fig. 9.4** Example of precision vs. robustness of trackers. The plot shows the tracking error of two stand-alone trackers with different observation models: maximum correlation (NCC), and randomized template tracking (RT). In this example, NCC is more accurate but fails early on, while RT is able to track over a longer period with less precision.

sequence when an observer's particular assumptions, e.g., slow motion or small pose change, do not hold. The number of tracked frames when running the tracker only once is dependent on when this event occurs: if it is near the beginning of a sequence the measured robustness is worse than when it is near the end. The performance of an observer is estimated as the expected error over all frames,

$$E[e^k] = \frac{1}{T} \sum_t e_t^k, \quad k = 1, \dots, K. \quad (9.3)$$

However, this function does not allow the comparison of observers when track is lost, because the error is meaningless in this case. In practice we are therefore interested in both the tracking error while the tracker is following the target as well as the probability of losing track. This motivates the distinction into two performance criteria, precision and robustness. Precision is related to the expected error only during successful tracking by

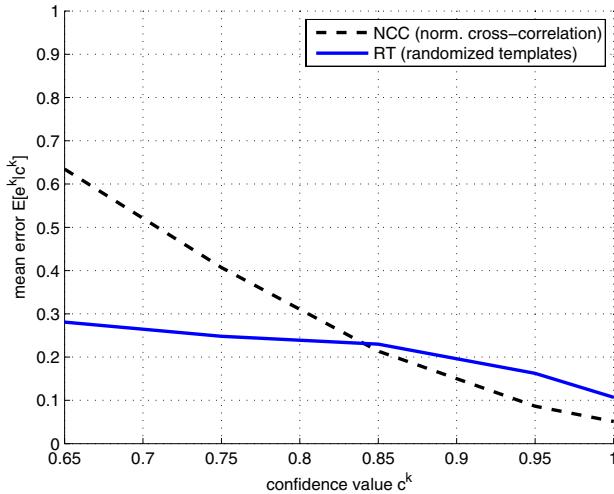
$$1 - E[e^k | e^k < \tau]. \quad (9.4)$$

The robustness is the probability of successful tracking as

$$E[e_t^k < \tau | e_{t-1}^k < \tau]. \quad (9.5)$$

See Figure 9.3 for a schematic of the evaluation on one sequence.

It is interesting to look at the relationship between precision and robustness. Observers with a fixed spatial model tend to be more precise than observers where the spatial arrangement is more flexible, see for example Figure 9.4, which shows tracking without re-initialization using single template matching (NCC) and local feature matching (RT) on one sequence. In this example NCC is more precise than RT, but the tracker loses lock earlier. Note that similar ideas have recently been explored in the visual object classification literature, where a representation's invariance vs. discriminative power trade-off was explored [80].

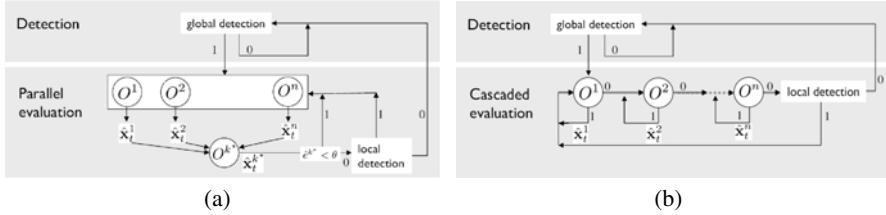


**Fig. 9.5** Expected error as function of confidence. This data is obtained from training sequences and allows the direct comparison of observers given their confidence values. Shown here are the results for observers NCC (normalized cross-correlation) and RT (randomized templates).

In order to evaluate each observer individually in a tracking algorithm, we use a threshold value of  $\tau = 1$  on the tracking error (in Equations 9.4 and 9.5) to determine loss of track. When this value is exceeded, the tracker is re-initialized at the next detection. The value of  $\tau = 1$  corresponds to the case where track has clearly been lost. Other threshold values could be used, where a smaller value enforces higher precision and lower robustness, and vice versa. Precision and robustness are then computed by taking expectations over all frames of the test sequences.

### 9.3.3 Evaluating Multiple Observers

This Section deals with the question of how to evaluate the performance that can be achieved by combining multiple observers. Ideally, we would like to select the observer  $O^k$  with the lowest error  $e_t^k$  at each time step. This information is not available at test stage, so instead the observer's confidence value  $c_t^k$  is used. Confidence values have often been used to compare the results of multiple observers and combine them [6, 9, 48]. However, most observers have a relatively simple object representation and thus the confidence value itself cannot be expected to be perfectly reliable. For example, an observer may have a high confidence value at an incorrect location if there is an object close-by that is similar to the target in the observer's feature space. The observer confidence values cannot be compared directly in our case, so they are simply regarded as features computed by each observer. In order to make



**Fig. 9.6** Evaluation schemes. (a) In the parallel evaluation, the output from the observer with the lowest expected error is chosen. (b) In the cascaded evaluation, the next observer is only evaluated if the expected error is above a threshold. An off-line trained detector is used to re-initialize. The binary tests in this schematic represent threshold tests on the expected error.

them comparable we estimate the distributions  $p(e^k | c^k)$  from the training data, i.e., the error distribution of observer  $O^k$  given its confidence value. To use the finite data set we discretize the range of the  $c^k$  values and compute  $p(e^k | c^k)$  in each partition. For the evaluation we represent it by the mean of each distribution,  $E[e^k | c^k]$ . Thus the estimated error for an observer  $O^k$  at time  $t$  is  $\hat{e}_t^k = E[e^k | c_t^k]$ . Figure 9.5 shows two of these functions for the normalized cross-correlation (NCC) and randomized templates (RT) observers. For example, if both observers return a confidence value of 0.9, the expected error of NCC is lower than that of RT.

We distinguish two different approaches of combining observers: parallel and sequential, respectively. In parallel evaluation, the estimates of multiple observers are available at each time step and the output of the most reliable observer is selected. Note that alternative fusion methods could be used, for example weighting the observer estimates. In sequential (or cascaded) evaluation observers are evaluated in sequence: If the first observer returns a high confidence, then no other observer is evaluated. Otherwise, the evaluation continues with the next observer. The advantage of sequential observation is that on average significantly less computation is required. However, the order of evaluation as well as the thresholds on the expected error are critical for good performance.

### 9.3.4 Parallel Evaluation

The parallel evaluation scheme selects the observer with the lowest expected error given its confidence value at each time step, i.e.,  $k_t^* = \operatorname{argmin}_k \hat{e}_t^k$ , see Figure 9.6(a). If this error is above a certain threshold, then the tracker is re-initialized at the next successful detection.

The running of tests consisting of all possible combinations of all trackers on all test sequences would be very time consuming. We therefore run all the observers individually on the test sequences and record the results on all frames. These are then used in the combination tests as the result from each component observer. In



**Fig. 9.7** Fist data set. The evaluation is performed on a data set of 12 sequences of 500 frames each, (top) six used for training and (bottom) six for testing. The sequences are taken with a static camera on a display showing people moving their hand in front of the screen. The dataset contains motion blur, skin-colored objects in the background, and occasionally other people in the scene.

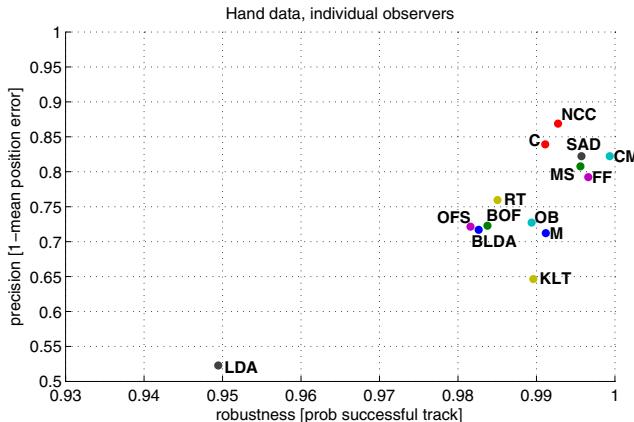
order to confirm the validity of such a set-up, we subsequently perform tests using the complete tracking framework for a few combinations of observers.

### 9.3.5 Cascaded Evaluation

Although the combined estimate is generally expected to be better than individual estimates, the main disadvantage is the increased execution time. In cascaded evaluation observers are evaluated in sequence, starting with the first observer, and continuing with the next observer only if the expected error is above a threshold value, see Figure 9.6(b). If no observer returns a sufficiently low expected error, the algorithm attempts to jump to the top of the cascade using local detection. In the evaluation, as in the parallel case, the output of the individual observers is used to estimate the performance of different combinations of observers as well as threshold values for switching observers.

### 9.3.6 Dynamic Model Discussion

A dynamic model is an integral component in every tracking algorithm as it can enable tracking through short periods of occlusion or weight the observations according to the most likely target motion. However, for the evaluation we aim to be independent of the dynamics, which are difficult to model in the case of rapid hand motion. Instead we sample the observation space densely at each pixel location in a neighborhood around the previous estimate and rely only on the observations without prediction, corresponding to a maximum likelihood location estimate. This methodology is consistent with the observation made in the particle filtering literature that the performance largely depends on the proposal distribution [22]. If good motion models are available, these should certainly be integrated in the final system [41]. The final tracking algorithm used in the gesture interface does employ prediction based on a constant velocity model for defining the search region and an auto-regressive filter for temporal smoothing.



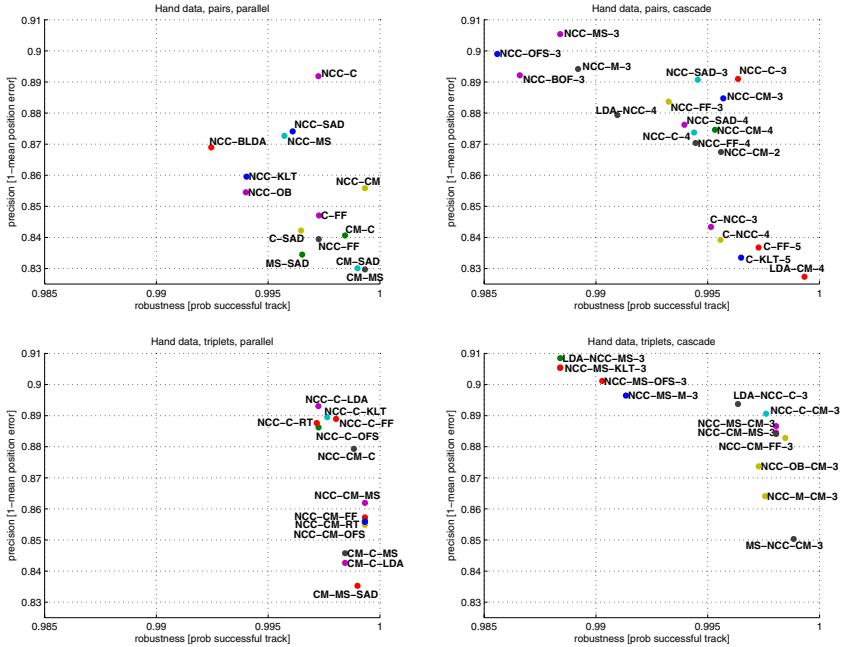
**Fig. 9.8** Evaluation of individual observers. This plot shows precision and robustness values on the test data. NCC is the most precise observer, the color-motion observer (CM) is the most robust.

### 9.3.7 Experimental Results

We evaluate the method on a hand dataset containing 12 sequences (10 with rapid motion, 2 with slower motion) of 500 frames each of size  $320 \times 240$ , recorded at 30 fps. The sequences are taken indoors with a static camera on top of a screen showing different people pointing their fist towards the camera in order to control a screen pointer. Frames of the dataset are shown in Figure 9.7. Half of the sequences are used to learn the expected errors  $E[e^k | c^k]$  for each observer  $O^k$ , the other half is used for performance evaluation.

### 9.3.8 Individual Observers

The precision and robustness measurements on the unseen test data are shown in Figure 9.8. A number of observations can be made. First, single template matching has high precision, with NCC being the most precise, and SAD the third most precise. Observers that include color also score highly, including the color probability (C), color-motion probability (CM), mean shift (MS) and flocks of features (FF) observers. Among the online classifiers, the online boosting (OB) observer shows the highest precision. Observers using local features generally perform slightly worse, with KLT and LDA observers ranking lower in terms of accuracy. In terms of robustness, the color-motion (CM) observer comes out on top, followed by the flocks of feature (FF) observer. Color based observers (MS, C) as well as single template observers (SAD, NCC) also perform well. Color and motion probability individually show similar robustness. The regular block-based optical flow



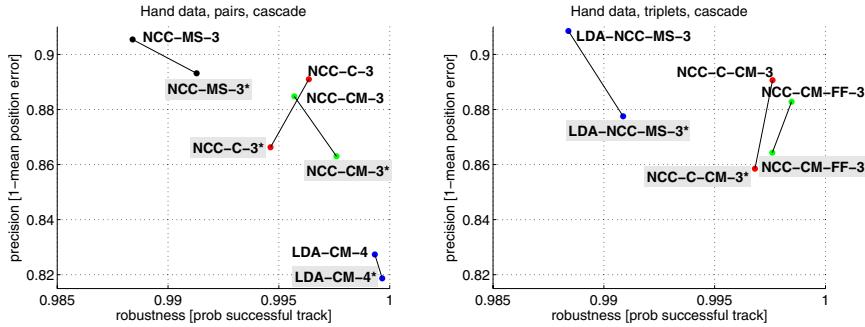
**Fig. 9.9** Evaluation of observer combinations. These plots show the precision and robustness measured on the test data. (top) pairs, (bottom) triplets, (left) parallel evaluation, (right) cascaded evaluation. Only a small subset of data points near the right upper corner with both high robustness and precision are shown in these graphs.

algorithm showed to be more robust than the KLT tracker, but both had difficulties handling rapid hand motion. The LDA observer shows significantly less robustness.

The performance of the off-line trained detector is not included in the evaluation. According to our definition of robustness it does not perform well because every missed detection is counted as loss of track. The percentage of correct detections on the test set is 48.4%, but it varies significantly across the sequences. On some sequences there are fewer detections due to blur and pose changes.

### 9.3.9 Observer Combinations

**Parallel Evaluation.** We evaluate all pairs of observers using a threshold value of  $\tau = 1$  on the expected error, resulting in a total of 91 combinations. Subsets of the results are shown in the top left of Figure 9.9. The graphs only show combinations that are near the right upper corner of high robustness and high precision. The combination of NCC with one of the color-based observers CM, C and MS shows good performance. In the videos the hand occasionally moves rapidly, resulting in significant motion blur. These cases tend to be failure modes for intensity or gradient



**Fig. 9.10** Comparison with real tracking results. These plots show the precision and robustness measured for selected combinations of observers. It compares the results by theoretical combination (as in Fig. 9.9) and real tracking results obtained for selected combinations of observers, shown here with gray background. The left plot shows the results on pairs and the right plot on triplets. The agreement is reasonable, although there is inherently some variation between the results.

based methods. On the other hand, the color distribution is less affected by motion blur. The robustness of these color-based observers is increased by most of the other observers that help to bridge the frames where the color cue is unreliable. The analysis also shows how observers using different cues complement each other. For example, the NCC-C combination has robustness-precision values of (0.997, 0.892), better than either NCC (0.992, 0.869) or C (0.991, 0.839) alone.

The evaluation of the 364 combinations of triplets shows a further improvement in performance, see the bottom left of Figure 9.9. Most noticeably, the best performance is achieved with combinations that include the NCC observer together with a color-based observer, C or CM. A local feature based observer such as LDA, KLT, or RT, can help too. Note, however, that the performance relative to the pairwise evaluation does not always change significantly. For example, by adding LDA to the NCC-C combination, the precision only increases slightly and robustness remains unchanged. Sometimes the precision can even decrease while robustness increases, such as in the case of NCC-C-FF. This means that on some occasions the additional observer helps to bridge gaps, but its estimate is otherwise not used.

**Cascaded evaluation.** We compared all ordered combinations of pairs at five different threshold levels (0.1, 0.2, 0.3, 0.4, 1.0) resulting in a total of 912 evaluations. Subsets of the results are shown in the top right plot in Figure 9.9. Most of the results with the highest precision employ NCC at the beginning of the cascade. High robustness is achieved when at least one of the observers uses the color cue, such as C or CM. The combination of NCC and CM (NCC-CM-3, i.e., a threshold value of 0.3) that was proposed in [74] performs well in terms of precision, only slightly worse compared to evaluating the same observers in parallel. Some combinations show higher precision, e.g., NCC-MS-3, however, this comes at the



**Fig. 9.11** Hand tracking results using NCC and color-motion (CM) observers. Shown are results of individual observers, a detector and parallel and cascaded evaluation. Colors indicate which estimate is used. In this sequence the hand is tracked successfully by both pair-wise schemes with a lower error than with either of the observers.

cost of lower robustness. It is also interesting to note that the performance of LDA in combination with other observers shows significantly improved robustness, e.g., LDA-CM-4, compared to its individual result.

We also evaluated all triplets of observers at five different threshold levels, a total of 4468 combinations. Subsets of the results are shown in the bottom right plot in Figure 9.9. As a general observation, the results are further improved. Successful combinations frequently include different types of observers, typically a single template, a color-based observer and either motion or local features. If one component is reliable over a long time period, the overall performance changes only little.

The results also suggest that in many cases arranging the observers in the order of their individual precision leads to good performance. Combinations that include NCC as first or second component perform consistently high. One idea is therefore to estimate using the most precise observer at each time step. If the expected error falls below the threshold, the next observer acts as a fallback method. Note that in some cases the cascaded tracker may have switched to an observer that is less precise during a difficult part of the sequence. It is therefore worth checking regularly if it is possible to jump to the top of the cascade again via local detection in order to increase tracking precision.



**Fig. 9.12** Hand tracking using NCC-CM-M observers in parallel. The NCC observer (blue) is used initially. During motion blur the tracker switches to the CM observer (red). For a couple of frames the M observer (purple) is used, while the light is turned off, before switching back to CM.



**Fig. 9.13** Hand tracking using NCC-CM-FF observers in a cascade. The NCC observer (blue) is used initially, switching to the CM observer (red) during motion blur.

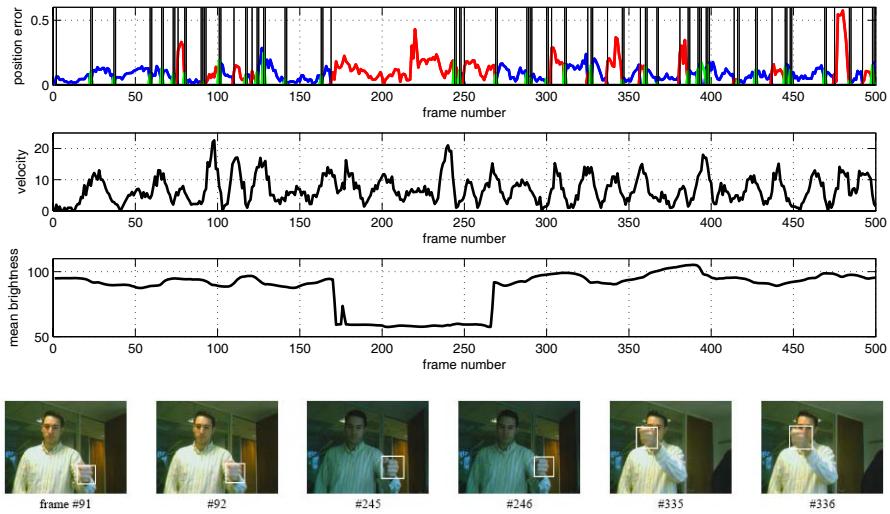
### 9.3.10 Tracker Evaluation on Selected Combinations

Given that the above analysis of observer combinations is based on the analysis of individual observers, an obvious question is how this result varies when the full combination is tested in a tracking framework. The two set-ups are not expected to give identical results because in the combined case the observer estimates are dependent on each other. Testing all combinations of observers becomes prohibitively expensive, thus we use the results on independent observations as a method to select promising combinations to evaluate. Figure 9.10 shows results on pairs and triplets using cascaded evaluation. On all examples the precision in the real tracking result is slightly smaller than to the results obtained with the simplified analysis, while the robustness values are very similar.

Figures 9.11 shows example results of two different observers individually, their pairwise combination, as well as the detector output. It can be seen that NCC is more precise, but in the end loses track due to fast motion. CM is less precise, but tracks the complete sequence successfully. The detector only fires in one frame in this example. Both pairwise schemes work well. In some cases the parallel and cascaded evaluation select different estimates, as in the second column of the figures.

Figures 9.12 and 9.13 show example frames from two test sequences using different observer triplets. Figure 9.12 shows results of the combination NCC-CM-M, evaluated in parallel. The NCC observer is used initially, but during fast motion the tracker switches to the CM observer. For a short while the motion (M) observer is used while the light is turned off. Figure 9.13 shows another case where the tracker switches from NCC to CM during fast motion.

The switching behavior of the NCC-CM tracker is illustrated in Figure 9.14, the same sequence as in Figure 9.12. During this sequence the light is turned off and



**Fig. 9.14** Switching trackers over time. This figure shows the tracker's switching behavior, colors in the plot indicate the component at each frame (blue=NCC, red=CM, green=detect). The hand velocity in pixels is shown in the second plot. During this sequence the light was turned off and on as can be seen in the mean brightness plot (third from top). Example frames where transitions occur are shown below (first and third pair from NCC to CM due to motion blur, middle pair from CM to NCC via local detection).

on, as shown in the mean brightness plot in Figure 9.12. As the light is switched off, the template used by NCC is no longer suitable and the tracker switches to the CM observer.

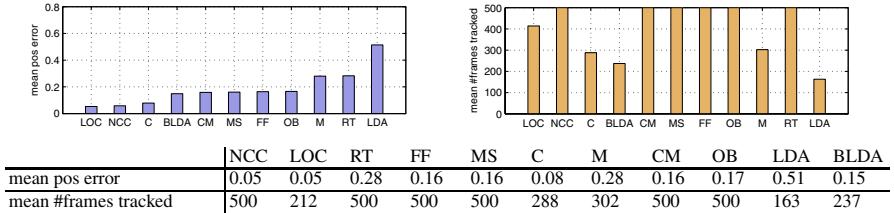
When examining the performance during slow object motion, it becomes clear that in these cases the NCC tracker has a very low error, while the color based trackers can be distracted by other skin-colored objects such as the arms. See Figure 9.15 for a comparison on two sequences with slow hand motion. In the comparison, local orientation correlation (LOC) matching [26] is included, which shows the same precision as NCC, but slightly lower robustness on this data set.

## 9.4 Gesture Interface System

This Section describes the components and the operation of the proposed gesture interface.

### 9.4.1 Visual Attention Mechanism

One goal of this work is being able to set up the system in an arbitrary environment, such as a living room or a public space, where multiple people may be within the



**Fig. 9.15** Results on two sequences with slow target motion. During slow hand motion the NCC and LOC observers which both use single templates are the most precise, however, LOC showed lower robustness.

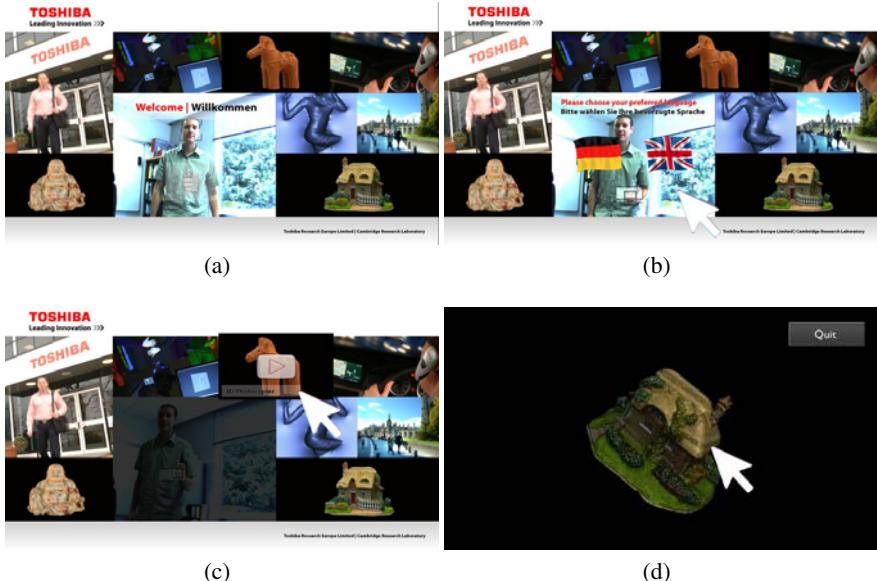
camera's view. For some periods there may be no interaction at all, until one person initiates the interaction in order to achieve a specific task. Initially, our system finds faces using a boosted detector [54]. Once a face is detected, the user is prompted to hold up their fist within a rectangular input area below their face, see Figure 9.16(a). This also works for multiple users in the scene as the input areas are ordered according to scale of the face detections, giving priority to users who are closer to the camera. When a fist is detected, an interaction area is defined, within which the fist is tracked. The area is placed at the detected fist location and is scaled proportional to the detected size, meaning that the range of hand motion is largely independent of the distance to the camera. In tracking mode the user is then able to browse content shown on the display.

#### 9.4.2 Tracking Mechanism

We take the results from the experiments in Section 9.3 which showed that an NCC-CM cascade gives good performance and use this as our fist tracker. The interface consists of a grid of windows, one of which shows the camera output. The 'active' region of fist tracking is shown as an overlay on the camera output window, see Figure 9.16(b). This window has the same aspect ratio as the overall screen, so the tracking result can be scaled up to give a cursor position on the screen, indicated with an arrow icon. The user can move their fist out of the interaction area at any time. In this case the tracker will fall back to global detection mode and re-calibrate the active tracking region to around the location of a newly detected fist.

Figure 9.17 shows the system with two users in the field of view. The system only allows interaction of one user at a time. Priority is given using a simple first-come-first-serve policy. The figure shows one user taking control first and as he drops his hand, the other user's fist is detected.

In order to ensure continuous tracking, two additional mechanisms are included in the system. The first is an additional local detection step if the NCC confidence value is too low. Instead of directly switching to CM, the tracker first attempts to re-detect and continue tracking with NCC. If local detection fails, it switches to CM. Secondly, there is a maximum number of frames that CM is used before detection



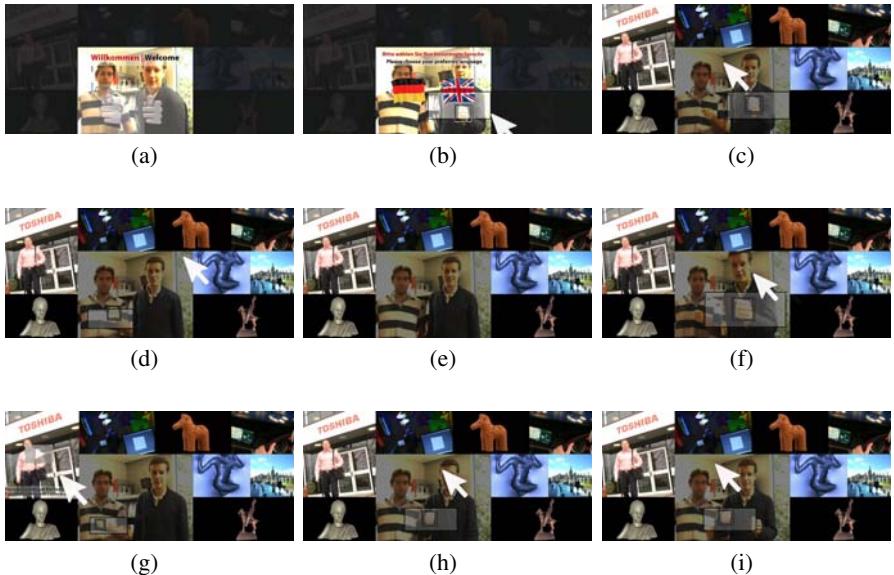
**Fig. 9.16** Gesture interface showcase. (a) During the attention phase, a face detector is run continuously and, as soon as a person looks towards the screen, a welcome message is displayed. (b) If the person holds up their fist within the shown rectangle placed below the detected face, tracking begins and the user can move a screen pointer. Shown here is the language selection screen. (c) Video content can be selected by hovering over the buttons that appear when the arrow is over the corresponding thumbnail image. (d) In the 3D model viewer application, the user can rotate a 3D model with their hand motion.

is triggered. Tracking with CM allows some variation in hand pose, but may also result in locking on to other skin colored objects. Correct tracking is verified with the detector, and if a fist is found, the system returns to the top of the cascade.

We use a Point Grey Flea2 camera, connected via a IEEE 1394b cable, to capture images of resolution  $320 \times 240$  pixels at 30 fps. The system has been implemented and tested on different platforms, including (i) a desktop with an eight-core Intel Xeon E5345, 2.33 GHz with 2GB RAM, (ii) a laptop with a dual-core Intel CPU T2600, 2.16 GHz with 1GB RAM. (iii) a laptop with a dual-core Intel Core 2 Duo T9800, 2.93 GHz with 3GB RAM. The system runs in real-time on these platforms and on the Core 2 Duo laptop it uses 30-50% of CPU cycles in tracking mode.

### 9.4.3 Selection Mechanisms

In order to activate a screen icon, we need to define a selection mechanism equivalent to a mouse click. Solutions that have previously been proposed include changing hand pose, finger or thumb extension, and simply hovering over an icon for a



**Fig. 9.17** Interaction example with two users. This figure shows the system’s behavior when two users are using the interface in turn. (a) Two faces are detected, both users are prompted to show a fist. (b) User on right raises hand, defining new interaction region, shown as a rectangle. (c) User on left shows fist, but user on right stays in control. (d) User on right lowers fist out of camera’s view, fist of user on left is detected. (e) Both users lower their hands, no fist detection. (f) When fist is shown closer to the camera, the interaction region becomes larger. (g) User on right lowers fist, control switches. (h) User on right takes over again, this time with his other hand. (i) User on left raises fist, user on right shows both fists, control stays with current hand.

short time period [11][26][37][45][52][67]. We have implemented these by training separate detectors, see Figure 9.18 (a) an open hand detector, (b) a ‘thumb up’ detector, and (c) hovering over an icon for a short period of time (0.5 seconds). Additionally, we propose the following method: (d) detecting a quick left-right shake gesture. The shake gesture is detected by recording the hand motion over a sliding time window of 20 frames and classifying this vector. In experiments linear discriminant analysis (LDA) and k-nearest neighbor classifiers were tested, but more reliable results were obtained by computing the distance to the closest positive training example (among a small set of 75 examples) and thresholding this value. The activation mechanism can be set according to the user’s preference, however, selection by hovering has been used as default setting during exhibitions.

A video can be played by selecting the button that appears when the cursor is over the corresponding thumbnail image, see Figure 9.16(c). Another sample application is a 3D model viewer, shown in Figure 9.16(d), where the user can rotate a displayed



**Fig. 9.18** Different gestures for selection. (a) Open hand pose, (b) thumb up pose, (c) hovering for a short time period, and (d) a shake gesture.

3D model with their hand motion. A video showing the system in operation can be viewed at <http://www.youtube.com/watch?v=RL9MpXhWCrQ&fmt=18>.

## 9.5 Summary and Conclusion

This Chapter has addressed the task of selecting component observers for particular tracking scenarios. To this end, a set of 14 observers has been evaluated on test sequences. A framework was proposed that evaluates the robustness and precision of observers, allowing the user to choose a profile suitable for a given application. The measurements of individual components were used to exhaustively evaluate combinations of components. We have shown results on observer pairs and triplets only, but the analysis can be applied to larger numbers of components.

The observers that were used in this paper have been used in stand-alone trackers. Some of these trackers themselves employ online feature selection. Here, instead of switching between relatively simple features from a pool [5, 18, 29], we propose switching online between observers that may use different cues and estimation schemes. Our evaluation framework allows combining arbitrary components that output an estimate and a confidence value. Direct comparison is possible because we estimate the observers' error distribution given their confidence.

In our experiments, cascaded evaluation gives similar performance to parallel evaluation at much higher efficiency. One suggested strategy is to use the most precise tracker if possible and use more robust ones as a fallback mechanism, with an off-line trained detector for re-initialization. This architecture allows for long term operation, which is required in many applications.

The proposed gesture interface works by tracking a pointing fist with a single camera facing the user. Our proposed system includes an attention mechanism that allows one user at a time to be in control. Note that face recognition could be employed for customizing the interface, as done in a previous version of our system [74]. For tracking the hand, we propose a multi-cue method that switches trackers over time and is updated continually by an off-line trained detector. Current sample applications include browsing videos as well as viewing a gallery of 3D models of sculptures. The system allows the user to view the 3D model from different directions by rotating it by hand. This can also be seen as a step towards

manipulation of virtual objects, which is still an active research area [7]. The system has been successfully used by over hundred people at conferences and public exhibitions.

## References

1. Argyros, A.A., Lourakis, M.I.A.: Real-time tracking of multiple skin-colored objects with a possibly moving camera. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3023, pp. 368–379. Springer, Heidelberg (2004)
2. Argyros, A.A., Lourakis, M.I.A.: Vision-based interpretation of hand gestures for remote control of a computer mouse. In: Huang, T.S., Sebe, N., Lew, M., Pavlović, V., Kölisch, M., Galata, A., Kisacanin, B. (eds.) ECCV 2006 Workshop on HCI. LNCS, vol. 3979, pp. 40–51. Springer, Heidelberg (2006)
3. Athitsos, V., Alon, J., Sclaroff, S., Kollios, G.: Boostmap: A method for efficient approximate similarity rankings. Boston University Computer Science Technical Report No. 2003-023 (2003)
4. Avidan, S.: Support vector tracking. IEEE Transaction Pattern on Analysis and Machine Intelligence 26(8), 1064–1072 (2004)
5. Avidan, S.: Ensemble tracking. IEEE Transaction Pattern on Analysis and Machine Intelligence 29(2), 261–271 (2007)
6. Badrinarayanan, V., Pérez, P., Le Clerc, F., Oisel, L.: Probabilistic color and adaptive multi-feature tracking with dynamically switched priority between cues. In: Proceedings of the International Conference on Computer Vision (2007)
7. Billinghurst, M., Kato, H., Poupyrev, I.: The MagicBook - moving seamlessly between reality and virtuality. IEEE Computer Graphics & Applications 21(3), 6–8 (2001)
8. Birchfield, S.: KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker, <http://www.ces.clemson.edu/~stb/klt/>
9. Birchfield, S.: Elliptical head tracking using intensity gradients and color histograms. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 232–237 (1998)
10. Black, M.J., Jepson, A.: Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In: Buxton, B.F., Cipolla, R. (eds.) ECCV 1996. LNCS, vol. 1065, pp. 329–342. Springer, Heidelberg (1996)
11. Bretzner, L., Laptev, I., Lindeberg, T.: Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In: Proceedings of the International Conference on Face and Gesture, pp. 423–428 (2002)
12. Buehler, P., Everingham, M., Huttenlocher, D.P., Zisserman, A.: Long term arm and hand tracking for continuous sign language tv broadcasts. In: Proceedings of the British Machine Vision Conference (2008)
13. de Campos, T.E., Murray, D.W.: Regression-based hand pose estimation from multiple cameras. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2006)
14. Canesta, <http://canesta.com> (Accessed on October 19, 2009)
15. Cipolla, R., Hadfield, P.A., Hollinghurst, N.J.: Uncalibrated stereo vision with pointing for a man-machine interface. In: Proceedings of the IAPR Workshop on Machine Vision Applications, pp. 163–166 (1994)
16. Cipolla, R., Hollinghurst, N.J.: Human-robot interface by pointing with uncalibrated stereo vision. Image and Vision Computing 14(3), 171–178 (1996)

17. Collins, R.T., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. *Transaction on Pattern Analysis and Machine Intelligence* 27(10), 1631–1643 (2005)
18. Collins, R.T., Zhou, X., Teh, S.K.: An open source tracking testbed and evaluation web site. In: *Proceedings of the International Workshop on Performance Evaluation of Tracking and Surveillance* (2005)
19. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. *Pattern Analysis and Machine Intelligence* 25(5), 564–575 (2003)
20. Cooper, H.M., Bowden, R.: Large lexicon detection of sign language. In: Lew, M., Sebe, N., Huang, T.S., Bakker, E.M. (eds.) *HCI 2007. LNCS*, vol. 4796, pp. 88–97. Springer, Heidelberg (2007)
21. Delamarre, Q., Faugeras, O.D.: Finding pose of hand in video images: a stereo-based approach. In: *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pp. 585–590 (1998)
22. Doucet, A., de Freitas, N.G., Gordon, N.J. (eds.): *Sequential Monte Carlo Methods in Practice*. Springer, Heidelberg (2001)
23. Du, W., Piatr, J.: A probabilistic approach to integrating multiple cues in visual tracking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) *ECCV 2008, Part II. LNCS*, vol. 5303, pp. 225–238. Springer, Heidelberg (2008)
24. Erol, A., Bebis, G., Nicolescu, M., Boyle, R.D., Twombly, X.: Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding - Special Issue on Vision for Human-Computer Interaction* 108, 52–73 (2007)
25. EyeToy, <http://www.eyetoy.com> (Accessed on October 19, 2009)
26. Freeman, W.T., Weissman, C.D.: Television control by hand gestures. In: *Proceedings of the International Workshop on Automatic Face and Gesture Recognition* (1995)
27. GestureTek, <http://www.gesturetek.com/> (Accessed on October 19, 2009)
28. de la Gorce, M., Paragios, N., Fleet, D.: Model-based hand tracking with texture, shading and self-occlusions. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2008)
29. Grabner, H., Bischof, H.: On-line boosting and vision. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 260–267 (2006)
30. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised on-line boosting for robust tracking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) *ECCV 2008, Part I. LNCS*, vol. 5302, pp. 234–247. Springer, Heidelberg (2008)
31. Graf, H.P., Cosatto, E., Gibbon, D., Kocheisen, M.: Multi-modal system for locating heads and faces. In: *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pp. 88–93 (1996)
32. Guan, H., Chang, J., Chen, L., Feris, R., Turk, M.: Multi-view appearance-based 3d hand pose estimation. In: *Proceedings of the International Workshop on Vision for Human Computer Interaction* (2006)
33. Hager, G.D., Belhumeur, P.N.: Real-time tracking of image regions with changes in geometry and illumination. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 403–410 (1996)
34. Hamer, H., Schindler, K., Koller-Meier, E., van Gool, L.: Tracking a hand manipulating an object. In: *Proceedings of the International Conference on Computer Vision* (2009)
35. Heap, A.J., Hogg, D.C.: Towards 3-D hand tracking using a deformable model. In: *Proceedings of the International Conference on Face and Gesture Recognition*, pp. 140–145 (1996)
36. Huttenlocher, D.P., Noh, J.J., Ruckridge, W.J.: Tracking non-rigid objects in complex scenes. In: *Proceedings of the International Conference on Computer Vision*, pp. 93–101 (1993)

37. Ike, T., Kishikawa, N., Stenger, B.: A real-time hand gesture interface implemented on a multi-core processor. In: Proceedings of the International Conference on Machine Vision Applications, pp. 9–12 (2007)
38. Ike, T., Kishikawa, N., Stenger, B.: A real-time hand gesture interface implemented on a multi-core processor. In: Proceedings of the International Conference on Machine Vision Applications, pp. 9–12 (2007)
39. Isard, M., Blake, A.: Condensation — conditional density propagation for visual tracking. *International Journal of Computer Vision* 29(1), 5–28 (1998)
40. Isard, M., Blake, A.: ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework. In: Burkhardt, H.-J., Neumann, B. (eds.) *ECCV 1998. LNCS*, vol. 1406, pp. 893–908. Springer, Heidelberg (1998)
41. Isard, M., Blake, A.: A mixed-state condensation tracker with automatic model-switching. In: Proceedings of the International Conference on Computer Vision, pp. 107–112 (1998)
42. Izadi, S., Agarwal, A., Criminisi, A., Winn, J., Blake, A., Fitzgibbon, A.: C-slate: Exploring remote collaboration on horizontal multi-touch surfaces. In: Proceedings of IEEE Tabletop (2007)
43. Jones, M.J., Rehg, J.M.: Statistical color models with application to skin detection. *International Journal of Computer Vision* 46(1), 81–96 (2002)
44. Kaucic, R., Perera, A.G.A., Brooksby, G., Kaufhold, J., Hoogs, A.: A unified framework for tracking through occlusions and sensor gaps. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 990–997 (2005)
45. Kölsch, M., Turk, M.: Fast 2D hand tracking with flocks of features and multi-cue integration. In: Proceedings of the International Workshop on Real-Time Vision for HCI (2004)
46. Kölsch, M., Turk, M.: Robust hand detection. In: Proceedings of the International Conference on Automatic Face and Gesture Recognition, pp. 614–619 (2004)
47. Krahnstoever, N., Schapira, E., Kettebekov, S., Sharma, R.: Multimodal human-computer interaction for crisis management systems. In: Proceedings of the International Workshop on Applications of Computer Vision, pp. 203–207 (2002)
48. Leichter, I., Lindenbaum, M., Rivlin, E.: A generalized framework for combining visual trackers – the black boxes approach. *International Journal of Computer Vision* 67(2), 91–110 (2006)
49. Li, Y., Ai, H., Yamashita, T., Lao, S., Kawade, M.: Tracking in low frame rate video: A cascade particle filter with discriminative observers of different lifespans. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
50. Lockton, R., Fitzgibbon, A.W.: Real-time gesture recognition using deterministic boosting. In: Proceedings of the British Machine Vision Conference, vol. II, pp. 817–826 (2002)
51. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 674–679 (1981)
52. MacCormick, J., Isard, M.: Partitioned sampling, articulated objects, and interface-quality hand tracking. In: Vernon, D. (ed.) *ECCV 2000. LNCS*, vol. 1843, pp. 3–19. Springer, Heidelberg (2000)
53. Microsoft Surface, <http://www.microsoft.com/surface/> (Accessed on October 19, 2009)
54. Mita, T., Kaneko, T., Stenger, B., Hori, O.: Discriminative feature co-occurrence selection for object detection. *Transaction on Pattern Analysis and Machine Intelligence* 30(7), 1257–1269 (2008)

55. Moreno-Noguer, F., Sanfeliu, A., Samaras, D.: Dependent multiple cue integration for robust tracking. Transaction on Pattern Analysis and Machine Intelligence 30(4), 670–685 (2008)
56. Nintendo Wii, <http://www.nintendo.com/wii> (Accessed on October 19, 2009)
57. Oblong Industries, <http://oblong.com/> (Accessed on October 19, 2009)
58. Oka, K., Sato, Y., Koike, H.: Real-time fingertip tracking and gesture recognition. Computer Graphics and Applications 22(6), 64–71 (2002)
59. Okuma, K., Taleghani, A., de Freitas, N., Little, J.J., Lowe, D.G.: A boosted particle filter: Multitarget detection and tracking. In: Pajdla, T., Matas, J.(G.) (eds.) ECCV 2004. LNCS, vol. 3021, pp. 28–39. Springer, Heidelberg (2004)
60. Ong, E.J., Bowden, R.: A boosted classifier tree for hand shape detection. In: Proceedings of the International Conference on Automatic Face and Gesture Recognition, pp. 889–894 (2004)
61. Ong, S.C.W., Ranganath, S.: Automatic sign language analysis: A survey and the future beyond lexical meaning. Transaction on Pattern Analysis and Machine Intelligence 27(6), 873–891 (2005)
62. Pavlović, V., Sharma, R., Huang, T.: Visual interpretation of hand gestures for human-computer interaction: A review. Transaction on Pattern Analysis and Machine Intelligence 19(7), 677–695 (1997)
63. Pérez, P., Vermaak, J., Blake, A.: Data fusion for visual tracking with particles. Proceedings of the IEEE 92(3), 495–513 (2004)
64. Playstation Eye,  
<http://www.us.playstation.com/ps3/accessories/scph-98047>  
(Accessed on October 19, 2009)
65. Project Natal, <http://www.xbox.com/en-us/live/projectnatal/>  
(Accessed on October 19, 2009)
66. Rehg, J.M.: Visual analysis of high dof articulated objects with application to hand tracking. Ph.D. thesis, Carnegie Mellon University, Dept. of Electrical and Computer Engineering (1995)
67. Robertson, P., Laddaga, R., Van Kleek, M.: Virtual mouse vision based interface. In: Proceedings of the International Conference on Intelligent User Interfaces, pp. 177–183 (2004)
68. Shimada, N., Kimura, K., Shirai, Y.: Real-time 3-D hand posture estimation based on 2-D appearance retrieval using monocular camera. In: Proceedings of the International Workshop RATFG-RTS, pp. 23–30 (2001)
69. Starner, T., Weaver, J., Pentland, A.: Real-time American Sign Language recognition using desk and wearable computer-based video. IEEE Transaction on Pattern Analysis and Machine Intelligence 20(12), 1371–1375 (1998)
70. Stefanov, N., Galata, A., Hubbald, R.: Real-time hand tracker using variable-length markov models of behaviour. Computer Vision and Image Understanding 108(1-2), 98–115 (2007)
71. Stenger, B.: Template-based hand pose recognition using multiple cues. In: Narayanan, P.J., Nayar, S.K., Shum, H.-Y. (eds.) ACCV 2006. LNCS, vol. 3852, pp. 551–560. Springer, Heidelberg (2006)
72. Stenger, B., Thayananthan, A., Torr, P.H.S., Cipolla, R.: Model-based hand tracking using a hierarchical bayesian filter. Transaction on Pattern Analysis and Machine Intelligence 28(9), 1372–1384 (2006)
73. Stenger, B., Woodley, T., Cipolla, R.: Learning to track with multiple observers. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2009)

74. Stenger, B., Woodley, T., Kim, T.K., Hernández, C., Cipolla, R.: AIDIA: adaptive interface for display interaction. In: Proceedings of the British Machine Vision Conference (2008)
75. Tomasi, C., Kanade, T.: Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University (1991)
76. Tosas, M.: Visual articulated hand tracking for interactive surfaces. Ph.D. thesis, University of Nottingham (2006)
77. Toshiba Qosmio Press Release,  
<http://laptops.toshiba.com/pressrelease/423413>  
(Accessed on October 19, 2009)
78. Triesch, J., von der Malsburg, C.: A system for person-independent hand posture recognition against complex backgrounds. IEEE Transaction on Pattern Analysis and Machine Intelligence 23(12), 1449–1453 (2001)
79. Ueda, N., Mase, K.: Tracking moving contours using energy-minimizing elastic contour models. In: Sandini, G. (ed.) ECCV 1992. LNCS, vol. 588, pp. 453–457. Springer, Heidelberg (1992)
80. Varma, M., Ray, D.: Learning the discriminative power-invariance trade-off. In: Proceedings of International Conference on Computer Vision (2007)
81. Viola, P., Jones, M.J.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137–154 (2004)
82. Wang, R.Y., Popović, J.: Real-time hand-tracking with a color glove. ACM Transactions on Graphics 28(3) (2009)
83. Wellner, P.: Interacting with paper on the digitaldesk. Communications of the ACM 36(7), 87–96 (1993)
84. Williams, O., Blake, A., Cipolla, R.: Sparse Bayesian learning for efficient visual tracking. Transaction on Pattern Analysis and Machine Intelligence 27, 1292–1304 (2005)
85. Woodfill, J., Zabih, R.D.: An algorithm for real-time tracking of non-rigid objects. In: Proceedings of the American Association for Artificial Intelligence (1991)
86. Wu, Y., Huang, T.S.: Vision-based gesture recognition: A review. In: Braffort, A., Gibet, S., Teil, D., Gherbi, R., Richardson, J. (eds.) GW 1999. LNCS (LNAI), vol. 1739, pp. 103–116. Springer, Heidelberg (2000)
87. Wu, Y., Huang, T.S.: View-independent recognition of hand postures. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 88–94 (2000)
88. Wu, Y., Huang, T.S.: Human hand modeling, analysis and animation in the context of human computer interaction. IEEE Signal Processing Magazine, Special issue on Immersive Interactive Technology 18(3), 51–60 (2001)
89. Wu, Y., Lin, J.Y., Huang, T.S.: Capturing natural hand articulation. In: Proceedings of the International Conference on Computer Vision, pp. 426–432 (2001)
90. Zhou, H., Huang, T.S.: Tracking articulated hand motion with eigen-dynamics analysis. In: Proceedings of the International Conference on Computer Vision, pp. 1102–1109 (2003)

# Chapter 10

## Multi-view Multi-object Detection and Tracking

Murtaza Taj and Andrea Cavallaro

**Abstract.** Multi-view trackers combine data from different camera views to estimate the temporal evolution of objects across a monitored area. Data to be combined can be represented by object features (such as position, color and silhouette) or by object trajectories in each view. In this Chapter, we classify and survey state-of-the art multi-view tracking algorithms and discuss their applications and algorithmic limitations. Moreover, we present a multi-view track-before-detect approach that consistently detects and recognizes multiple simultaneous objects in a common view, based on motion models. This approach estimates the temporal evolution of objects from noisy data, given their motion model, without an explicit object detection stage.

### 10.1 Introduction

Object detection and tracking is a fundamental task in various video-based applications such as security, sport analysis and tele-collaboration. Because occlusions and limited field of view make detection and tracking a challenging task, multiple video cameras can be used to increase the observability of objects, thus facilitating their consistent identification over time. Multi-camera tracking aims to establish the spatio-temporal correspondence of the same object across multiple views.

The modeling of the multi-view tracking problem depends on the management policy, the network type and the coverage of the network. The management policy of a network can be centralized [1], distributed [2] or hybrid [3]. The network can be composed of passive cameras [1], active cameras [4], or a combination of both.

---

Murtaza Taj

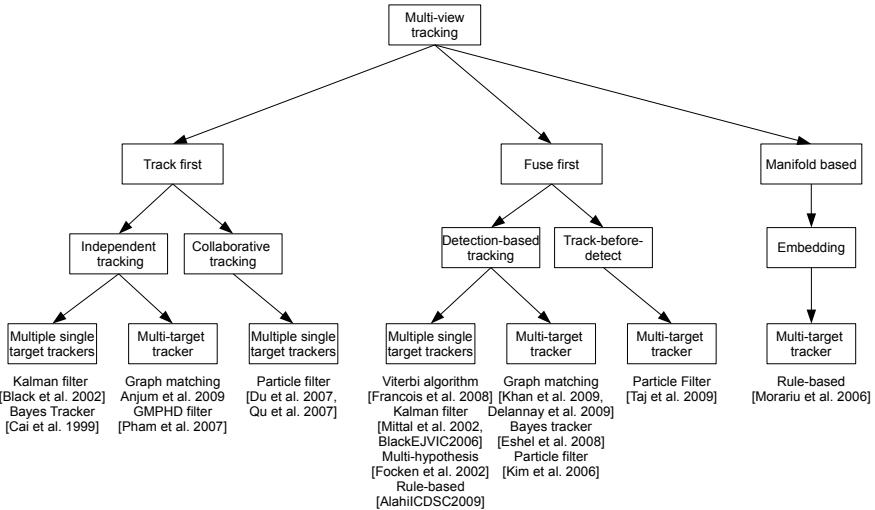
Queen Mary University of London, UK

e-mail: [murtaza@elec.qmul.ac.uk](mailto:murtaza@elec.qmul.ac.uk)

Andrea Cavallaro

Queen Mary University of London, UK

e-mail: [andrea.cavallaro@elec.qmul.ac.uk](mailto:andrea.cavallaro@elec.qmul.ac.uk)



**Fig. 10.1** Overview of multi-view tracking approaches.

As for the coverage of the network, the cameras can have partially overlapping [5] (multi-view) or non-overlapping [6] fields of view. This Chapter will focus mainly on tracking algorithms for partially-overlapping passive camera networks working with a centralized management policy, although some algorithms could be extended to work in a distributed fashion or with active cameras.

Algorithms for target tracking in multi-view camera networks can be grouped based on the modalities for tracking and information fusion and can be categorized into three main classes, namely *track-first*, *fuse-first* and *manifold-based*. The categorical overview of these approaches is shown in Figure 10.1. Track-first approaches perform tracking in each camera view and then project and link the resulting information on other views. Fuse-first approaches project detection information from each view to a common view and then apply tracking. Track-first approaches are in general more complex computationally but require a lower data transfer load. Track-first and fuse-first classes will be discussed in details in the rest of the Chapter.

Manifold-based approaches can be used when camera calibration information is not available, cannot be computed efficiently, or the assumption that the world is planar is not applicable. In this category, multi-camera tracking can be performed by projecting features on a manifold through Locally Linear Embedding [7]. The approach uses Caratheodory-Fejer (CF) interpolation theory, which is robust against model uncertainty and occlusion, to identify the dynamic evolution of the data on the manifolds. This method assumes that multiple views are highly overlapping and uses rule-based multi-target tracking with multiple hypotheses. The approach relies heavily on the training that uses segmented foreground objects.

The Chapter is organized as follows. The problem of multi-camera tracking is formulated in Section 10.2. Section 10.3 discusses the calibration and data fusion

between multiple views using plane-to-plane homography. Track-first approaches are discussed in Section 10.4 that covers methods using independent trackers and collaborative trackers. Fuse-first approaches are described in Section 10.5 that covers detection-based tracking as well as simultaneous detection and tracking. Finally, in Section 10.6 we draw the conclusions.

## 10.2 Problem Formulation

Let a wide area be monitored by a set  $C = \{C_1, \dots, C_c, \dots, C_N\}$  of  $N$  cameras. Let  $\mathbf{x}_k^{c,i}$  be the state of the  $i^{\text{th}}$  object in camera  $C_c$  and let  $\mathbf{x}_k^{\pi,i}$  be the state of the  $i^{\text{th}}$  object on the common view plane  $\pi$ .

The state  $\mathbf{x}_k^{c,i}$  can be defined based on a set of features, such as the position and the velocity components of the target in the image plane, the width and the height of the bounding box (or the axes of the ellipse) defining the area of the target, and a representation of the appearance (such as the color histogram) of the target [8].

As mentioned in Section 10.1, the multi-camera tracking problem can be categorized into two classes, namely track-first or fuse-first.

- *Track-first* approaches can be divided into four steps:

1. Target *localization* in each view. This step extracts the localization information or measurement  $Z_k^c = \{z_m^c | m = 1, \dots, k\}$  in each view.
2. Target *state estimation*,  $\mathbf{x}_k^{c,i}$ , in each view, given the set of measurements  $Z_k^c$  up to time  $k$  and the state  $\mathbf{x}_{k-1}^{c,i}$  at previous time  $k - 1$ .
3. State estimates *projection* to a common view (from individual views). This step projects the tracks from the image views to a common view  $\pi$ , using the projection matrix  $\mathbf{H}^{c,\pi}$ , which performs a mapping from camera  $C_c$  to  $\pi$ :

$$\mathbf{x}_{1:k}^{\pi,i} = \mathbf{H}^{c,\pi} \mathbf{x}_{1:k}^{c,i}, \quad (10.1)$$

where  $\mathbf{x}_{1:k}^{\pi,i}$  is the projection of track  $\mathbf{x}_{1:k}^{c,i}$  from camera  $C_c$ . Note that  $\pi$  can be the camera view selected as reference view [9, 10] or a hypothetical top view [11, 12, 13, 14].

4. *Correspondence* resolution between projections from multiple views. This step establishes the link between all the tracks  $\mathbf{x}_{1:k}^{\pi,i}$ , projected from different views, belonging to the same object. The fused tracks can be reprojected to the individual views for improving track estimates.

- *Fuse-first* approaches can be divided into three steps:

1. Target *localization* in each view. This step extracts the localization information or measurement  $Z_k^c = \{z_m^c | m = 1, \dots, k\}$  in each view.
2. *Projection* of localization features from each view to a common view. This step projects the localization information or measurement to  $\pi$ :

$$Z_k^{\pi,c}(u, v) = \mathbf{H}^{c,\pi} Z_k^c(x, y); \quad (10.2)$$

and fuses them:

$$Z_k^\pi(u, v) = \zeta(\{Z_k^{\pi,c}(x, y)\}_{c=\{1, \dots, N\}}), \quad (10.3)$$

where  $\zeta$  is a function that fuses the measurements from multiple cameras.

3. *State estimation* in the common view. This step estimates the state  $\mathbf{x}_k^{\pi,i}$  of each object in  $Z_k^\pi$ . The state can be estimated using traditional detection and tracking schemes or via simultaneous detection and tracking.

## 10.3 Calibration and Fusion

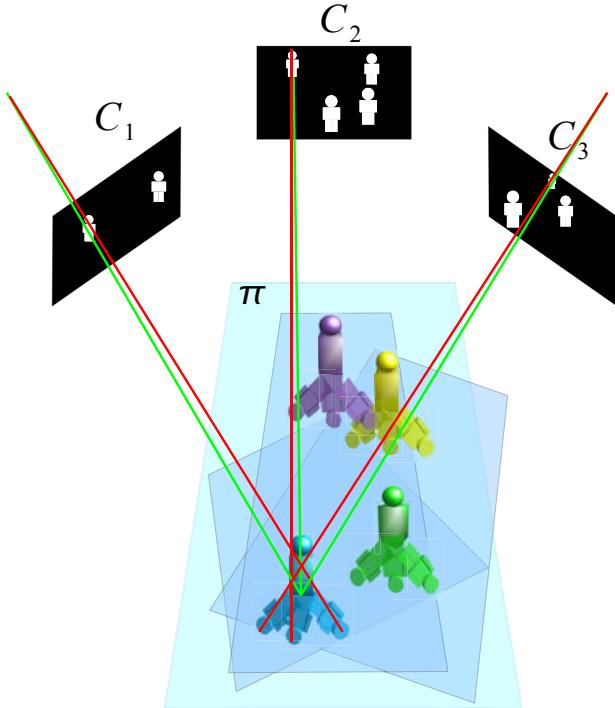
To fuse the data from multiple views, track-first and fuse-first approaches assume the availability of camera calibration information. Homographic transformation matrices are generally used to this end. Homographies can be computed manually [15] or automatically [16] by identifying corresponding points between views.

The automatic selection of corresponding points (auto-calibration) may be obtained through trajectory correspondence, field-of-view lines or feature-point correspondence. *Trajectory correspondence* can be achieved with a least mean square search on trajectory points from multiple views [10] or by using features such as position, velocity, size and color [17]. Automatically recovered *field-of-view lines* use the correspondence between objects in multiple views when they enter or exit the scene (i.e., when they appear on field-of-view lines in overlapping views). Both trajectory-based and field-of-view-lines-based approaches rely heavily on detection and tracking performance and assume that reliable tracks are available from each camera. Auto-calibration can also be performed using *feature-point correspondence*, for example using SIFT features followed by RANSAC to reject outliers [16]. The limitation of this approach is the assumption that the ground plane in each view is sufficiently textured in order to facilitate a reliable point correspondence.

The calibration information can then be used to map information from one view to another, using single or multi-level homography.

### 10.3.1 Single-Level Homography

Different features, such as points or segmentation masks, can be projected on the common view. In case of *point* projection (e.g., feet location [5, 18] or blob centroid [19]) a binary signal identifies the points (Figure 10.2), thus making this approach very sensitive to detection errors in a view. Although the error can be reduced with a Gaussian Kernel on the common view [20], these approaches are not applicable in crowded scenes, as feet or centroid locations may not be visible or may be misleading due to occlusions. In crowded scenarios a preferred solution is to track head locations that can be obtained by projecting the whole information represented by the change *segmentation mask*. In this case,  $Z_k^\pi(u, v)$  can be obtained by computing the variance at each pixel [21] as



**Fig. 10.2** Projection of the detections from multiple views to the top view.

$$Z_k^\pi(u, v) = \frac{1}{\sigma^2(\{Z_k^{\pi,c}(u, v)\}_{c=1}^{N_c})}, \quad (10.4)$$

where

$$Z_k^{\pi,c}(u, v) = \begin{cases} \mathbf{H}^{c,\pi} Z_k^c(x, y) & \text{if } \bar{Z}_k^c(x, y) = 1 \\ 0 & \text{otherwise} \end{cases}. \quad (10.5)$$

$\bar{Z}_k^c(x, y)$  is the foreground binary mask value at  $(x, y)$  in  $C_c$  that is projected to  $(u, v)$  in  $Z_k^{\pi,c}$ , a single channel image.

Similarly, instead of the actual pixel values [22, 23], the binary mask values can be projected:

$$Z_k^\pi(u, v) = \sum_{c=1}^{N_c} Z_k^{\pi,c}(u, v), \quad (10.6)$$

where

$$Z_k^{\pi,c}(u, v) = \begin{cases} 1 & \text{if } \bar{Z}_k^c(x, y) = 1 \\ 0 & \text{otherwise} \end{cases}. \quad (10.7)$$

As the aforementioned approaches use a foreground binary mask, they perform prior thresholding on the image plane and may therefore ignore low-contrasted or small targets.

An alternative approach is to project the *motion estimation likelihood* without any thresholding. In this case, one can compute on the common view the product of the likelihood values from each camera [16]:

$$Z_k^\pi(u, v) = \prod_{c=1}^N Z_k^{\pi,c}(u, v), \quad (10.8)$$

where  $Z_k^{\pi,c}(u, v)$  is the projected likelihood value. The drawback of this approach is that, instead of just the foreground pixels, the entire likelihood image from each view has to be projected for each time  $k$  on the common view, thus increasing the computational load.

Finally, as target points or features from more than one view can be projected on the same pixel position on the common view, each point  $(u, v)$  in  $Z_k^\pi(u, v)$  has to be normalized with respect to the number of overlapping cameras in that region.

### 10.3.2 Multi-Level Homography

To increase the amount of discriminative information in the projection, one can compute the homography from multiple planes that are parallel to the ground plane [24]. Such homographies can be obtained by moving along the vertical vanishing points and then estimating projection planes that are parallel to the planar top view [16].

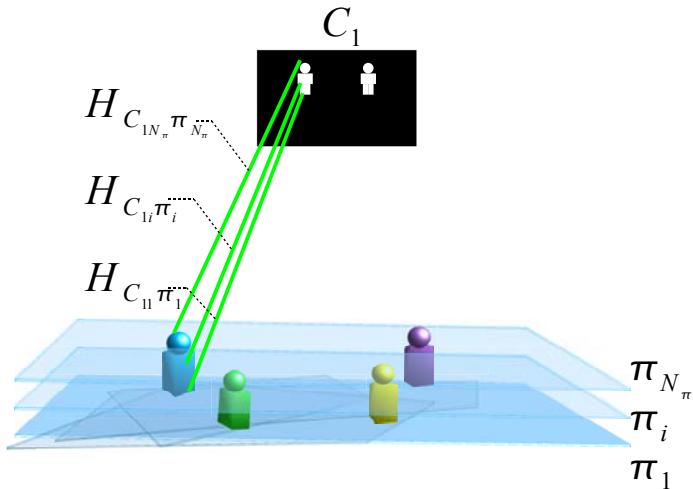
Let  $\mathbf{H}^{c_j\pi_j}$  be the homographic matrix that projects points from  $c_j$ , the  $j^{th}$  plane in the camera  $C_c$ , to the  $j^{th}$  common-view plane  $\pi_j$  as

$$Z_k^{\pi_j}(u, v) = \mathbf{H}^{c_j\pi_j} Z_k^{c_j}(x, y). \quad (10.9)$$

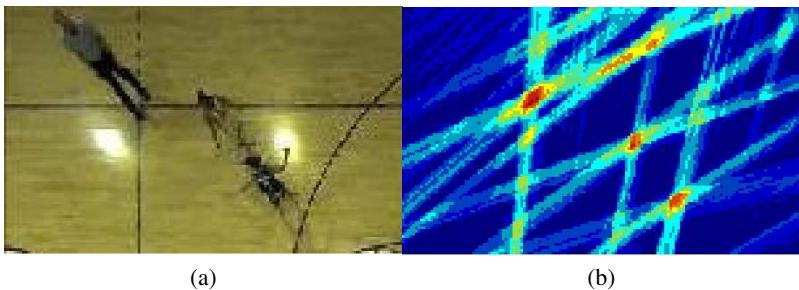
The projections on multiple planes can either be treated separately to obtain the information about the object shape [16] or can be combined as mentioned for the single-level homography by concatenating the feature vectors from each parallel plane [22]. Figure 10.3 shows an illustration of the projection of the localization information from a camera view to multiple planes on the common view. The common view can be generated through the fusion of the pixel values from three homography planes, one at the feet level, one at the head level and one between these two planes. The fusion of the pixel values can be performed using Equation 10.4 that creates a variance map.

The signal intensity at each position is proportional to the number of foreground pixels being projected onto that position. In a multi-level homography, pixels representing different portions of an object (e.g., a person) in the image view along the vertical-axis (e.g., feet, legs, torso, neck and head) are projected around the same position on the common view, thus increasing the signal intensity.

The signal strength depends upon the number of cameras observing that region, as points contributed from multiple cameras are projected on the same location on



**Fig. 10.3** Detections projected from one view to multiple parallel planes.

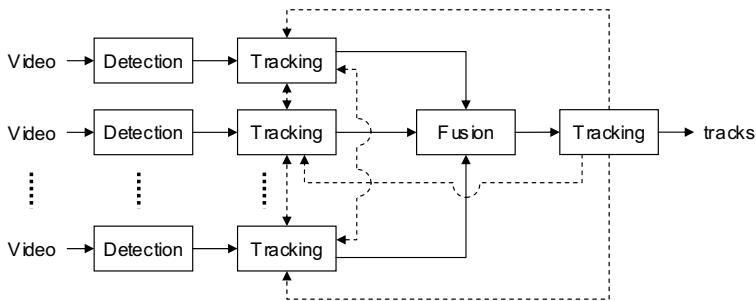


**Fig. 10.4** Example of parallax error. (a) Top view with 3 targets. (b) 3 high-intensity regions on the top view generated by the projections of the targets together with several other high-intensity regions due to *phantoms*.

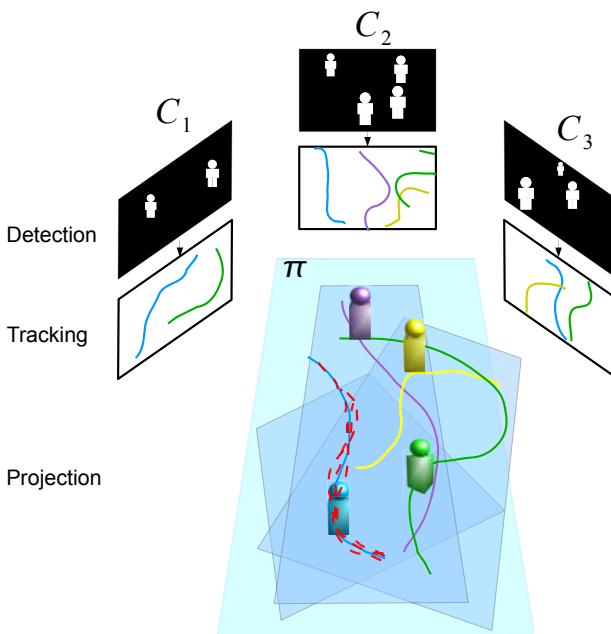
the common view. This compensates for possible miss-detections in some camera views. However, when an object is projected on the plane, pixels not belonging to that plane are also projected there, thus creating a *shadow* of the object along that plane. These projected shadows (from multiple objects) can overlap with each other and create false signal intensities. These noise components, referred to as *parallax errors* [22] or *phantoms* [25] (Figure 10.4), have to be filtered by the tracking algorithm, as discussed in the next sections.

## 10.4 Track-First Approaches

Track-first multi-view tracking can be performed either *independently* in each view or *collaboratively* across views. In collaborative tracking, estimated tracks in the



**Fig. 10.5** Generic block diagram of track-first multi-view tracking algorithms. Solid line: independent tracking. Solid and dotted line: collaborative tracking.



**Fig. 10.6** Illustration of the track-first approach.

image view and in the common view can be used to assist each other and to improve track estimates in one view (Figure 10.5). Both independent and collaborative algorithms first track objects in each camera view and then project the tracks onto the common view for fusion (Figure 10.6). The problem to be solved here is the fusion of the multiple tracks belonging to the same target.

### 10.4.1 Independent Tracking

Independent tracking computes the projection of single-view tracks to another camera view [10] or to the hypothetical top view [11].

Kalman filter state estimates on the image plane can be used for single-target tracking on the top view using a second Kalman filter based on covariance mapping [10]. For multi-target tracking, independent tracks from each view are projected on the common view or on the top view for fusion. The challenge is that multiple corresponding tracks may not overlap with each other in time and space. In fact, targets may be visible in one camera during a certain time interval and in another camera during another interval. This problem can be solved by trajectory association using multiple spatio-temporal features with an off-line processing that allows recovering from failures due to occlusions and target merging [11].

A Gaussian Mixture PHD filter (GMPHD) can be used for on-line multi-target tracking using independent trackers [13]. GMPHD can be applied on each view as well as on the top view for track estimation, using features such as position, size and color histograms. The 2D estimates of the target state from each view can be projected onto the top view and used as observations for the GMPHD filter. Tracking can be performed by assigning a label to each Gaussian component. Approaches based on the PHD filter are computationally efficient as the complexity increases only linearly with the number of targets.

As estimates in a view can be affected by partial occlusions, the drawback of independent tracking is that the tracking in one view does not help improving tracking results in another view. An alternative solution is to perform collaborative tracking by using track estimates from a view as measurement for other views, as discussed in the next section.

### 10.4.2 Collaborative Tracking

In collaborative tracking, a set of measurements from a view are used to improve tracking results in other views.

Objects can first be tracked using a particle filter in each view and then the particles can be projected onto the top view for fusion [12]. To compute the precise location of the target on the top view, the principle axis<sup>1</sup> of the target can be defined in each view and then projected on the top view. The intersection of the projected principle axes can be used as the target location. The closeness of the particle to the principle axis is used as the likelihood criterion in the particle filter. To improve the results on individual views using top-view tracking, the particles in each view can be sampled from both camera-view particles and top-view particles [12].

Similarly, multiple independent regular particle filters (MIPFs) can be used to track each target in a view. The posterior in each camera can be computed by using

---

<sup>1</sup> The principle axis is the vertical line from the bottom (e.g., the feet of a person) to the top (e.g., the head of a person) of a target.

**Table 10.1** Track-first multi-camera tracking algorithms. (Key: GMPHD = Gaussian Mixture Probability Hypothesis Density; MT = Multi-target tracker; IT = Independent tracking; CT = Collaborative tracking; M = Manual)

	Ref.	Features	Tracker	Calib.	MT
IT	[10]	2D position, size, velocity	Kalman filter	M	No
	[9]	2D position, height and intensity	Bayes tracker	M	No
	[11]	2D position, size, velocity	Graph matching	M	Yes
	[13]	position, size and color histogram	GMPhD filter	M	Yes
	[26]	2D position	Template matching	M	Yes
CT	[12]	2D position, size	Particle filter	M	No
	[14]	5D state space using ellipses	Particle filter	M	No

the measurements from all the cameras [14]. A summary of state-of-the-art track-first approaches is shown in Table 10.1

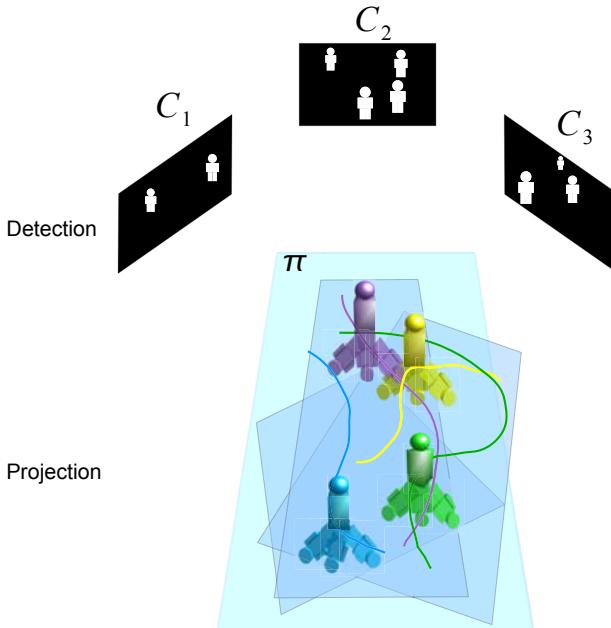
Track-first approaches involve multiple tracking steps and hence can be computationally expensive. To reduce the complexity, fuse-first approaches can be used that defer the tracking step until when the information from each view is fused on a common view.

## 10.5 Fuse-First Approaches

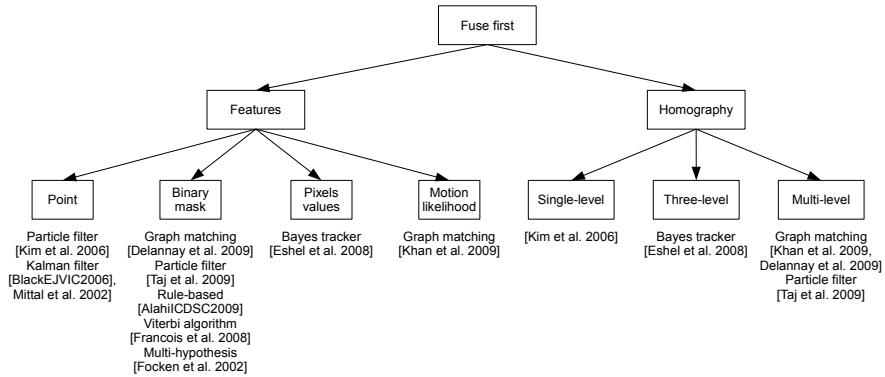
Although collaborative track-first approaches help improving trajectory estimation in each camera view, they involve multiple tracking steps that can introduce sources of estimation error. These multiple steps can be eliminated by tracking on the common view only, by accumulating on the common view the information from each view (Figure 10.7).

Fuse-first multi-view tracking approaches are characterized by the features used and by the strategy for the computation of the common view (Figure 10.8). The features extracted can be the feet location of people [5], the silhouette centroid [19], the change segmentation mask [22], the foreground pixels or the whole motion segmentation likelihood [27].

Note that although fuse-first methods involve one tracking step only, they may involve multiple *detection* steps: (i) in each camera view, before fusion and (ii) on the common view, after fusion. Furthermore, as the fusion involves triangulation of noisy information, this can result in a larger number of solutions (i.e., candidate targets) than desired. To address this type of data and to reduce the overall complexity of the tracker, simultaneous detection and tracking can be performed that does not require a detection step (Figure 10.9). The various aspects of these multi-view tracking techniques are discussed in this section. A summary of the state-of-the-art of fuse-first multi-view tracking approaches is shown in Table 10.2.



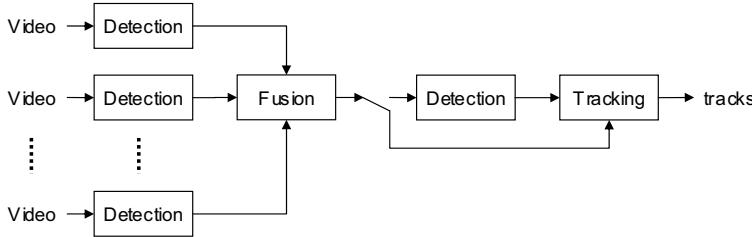
**Fig. 10.7** Illustration of the fuse-first approach.



**Fig. 10.8** Fuse-first multi-view tracking approaches: features and homography.

### 10.5.1 *Detection-Based Tracking*

Detection-based trackers first localize objects on the common view and then track them. Target localization (detection) can be performed by thresholding [21, 23] or by quantizing the top view into a grid such that each sub-area can only contain one target. A dictionary of atoms modeling the presence of an object at a given

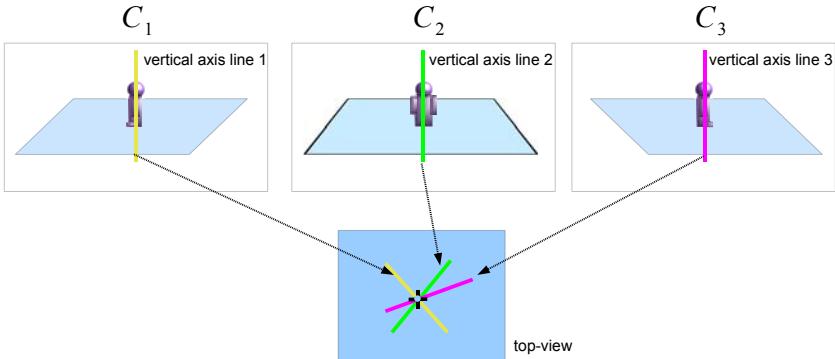


**Fig. 10.9** Generic block diagram of fuse-first multi-view tracking algorithms. The switch differentiates between detection-based trackers and simultaneous detection & tracking algorithms that do not require the detection step after fusion.

**Table 10.2** Fuse-first multi-view tracking algorithms (MT: Multi-target tracking; MHPT: Multi-Hypothesis Probabilistic Tracker; M = Manual; A = Automatic)

Ref.	Features	Tracker	Calib.	MT
[28]	color and motion	Viterbi algorithm	M	Yes
[5]	person vertical axis, ground position	Particle filter	M	Yes
[20]	feet position	Kalman filter	M	Yes
[19]	color histogram, bounding box, centroid	MHPT	M	Yes
[21]	head position	Bayes tracker	M	Yes
[16]	multiple planes occupancy map	Minimum graph cut	A	Yes
[27]	field of view lines	not mentioned/any	M	NA
[22]	foreground mask	Particle filter	M	Yes
[29]	foreground mask	Rule-based	M	No
[23]	foreground mask	Graph cut	M	Yes
[30]	2D position	Kalman filter	M	No

location in a view can then be used to identify if the position in the quantized top-view grid contains a target [29]. When the top view is composed of projected points representing target centroids or feet locations, all the non-zero values can be used as candidate locations [30]. A ray can be drawn from the center of projection of each camera through the centroid of foreground regions from that camera. The intersection of these rays can then be used as target location, which can be tracked using Multi-hypothesis Probabilistic Tracker (MHPT) [19]. Similarly, the vertical axis of the target across views can be mapped on the top-view plane and their intersection point on the ground can be used as the feet location of the target on the top view [5] (Fig. 10.10). Contrary to [12], in [5] target feet locations (obtained through vertical axis lines) are not tracked in each camera view but detected and tracked only once on the common view. The detection step involves associating multiple projected points to the same target by using a threshold on the inter-point distance. The top-view feet locations can then be tracked using a single-view tracker such as particle filter. The thresholding on inter-point distance for associating multiple projected feet locations belonging to the same target can be eliminated by using a Gaussian kernel



**Fig. 10.10** Illustration of the target vertical axis intersection on the top view.

for a single image pair [20]. This results in a common plane that is similar to the one generated using foreground masks (Equation 10.4, Equation 10.6) or likelihood maps (Equation 10.8).

When the common view is based on foreground masks, object segmentation can also be performed by thresholding, thus resulting in a large number of points for each target. These points need to be grouped to obtain the target location. The grouping can be performed using K-means, Mixture of Gaussians or Mean-shift [31, 32]. The mean of these clusters represents the target location, which can be tracked using single-view trackers such as multiple single target Kalman filter [20]. Color and motion information can also be used in the common view with a generative model to explicitly handle complex occlusions and interactions between objects [28]. The tracking of each object can be performed using the Viterbi algorithm. A greedy approach that makes the locally optimal choice at each stage can be used to avoid the combinatorial explosion of the computational cost due to joint posteriors. Unlike approaches that perform state estimation using frame-to-frame correspondence only, this method computes global optima of scores summed over several frames, thus making it more robust to persistent and prolonged occlusions. However, this approach can only process a batch of frames at a time and hence the results are delayed.

To further improve the effectiveness of tracking in the fused domain, multi-level homography can be used [24] (see Section 10.3.2). Head detection can be performed by thresholding the variance map (Equation 10.4) and by employing floor-level homographic projections. Finally, the candidate head-top positions can be estimated by clustering with double threshold hysteresis. Note that head tracking requires the cameras to be mounted at a significant height so that the heads are fully visible. The number of homography levels can be increased to further improve the localization information [16], at an additional computational cost. The localization information can also be improved by projecting the motion segmentation likelihood values and obtaining the mask by taking the product of the values from multiple views (Equation 10.8). The foreground likelihood probabilities from each plane of each

view at each time can be projected onto the corresponding plane of the common-view to obtain a 4D spatio-temporal occupancy map. The minimum graph cut procedure can be applied with alpha-expansion to segment targets. Trajectory segmentation can then be performed using graph cut. Although this approach shows promising results, it is computationally very expensive as it requires obtaining a 4D occupancy map before applying the minimum cut procedure.

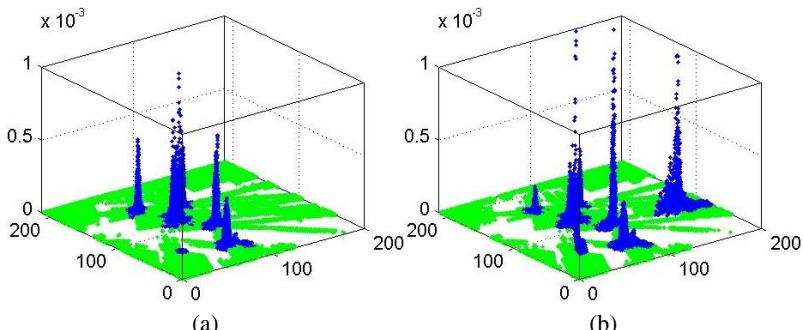
To reduce the computational cost and to obtain an on-line solution, multiple homography planes can be collapsed on the ground plane [23]. Similar thresholding and clustering can be performed to localize targets followed by graph matching to obtain the tracks.

The thresholding step to localize targets is a bottleneck in most detection-based trackers. Furthermore, due to parallax error (Figure 10.4), false peaks can be selected as candidate target locations. These false peaks can be filtered using heuristics on size and speed [25]. A better alternative is to perform tracking without applying the detection step by using simultaneous detection and tracking via track-before-detect.

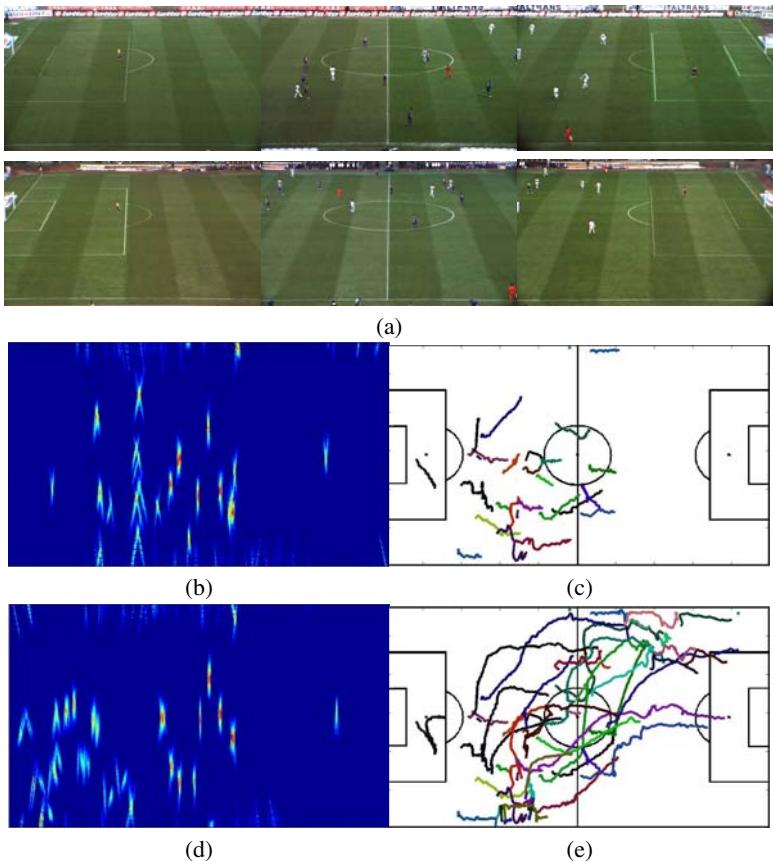
### 10.5.2 Track-Before-Detect

*Track-before-detect* (TBD) is a Bayesian approach that extends the target state with the signal intensity and evaluates each image segment against a certain dynamic model. As the target intensity along with its dynamics follow a statistical model, this approach allows us to track targets with lower signal strength, without applying an additional detection step. The state estimation can then be performed using particle filtering [22].

To avoid an explicit target localization step, in track-before-detect the entire input signal is considered as a measurement. This measurement is a highly non-linear function of the target state and can be solved either by discretization of the state [34]



**Fig. 10.11** Example of particle weights and positions. (a) Without multi-target update (one target has very small weights and another one is missing); (b) with multi-target update. As the weights for weak targets are very low, without the multi-target update strategy, lost tracks are possible.



**Fig. 10.12** Multi-view tracking on the top view on frames 160 and 500 of ISSIA dataset. (a) Original frames from each view. (b,d) Top view after fusion. (c,e) Tracks generated with multi-target track-before-detect.

or by non-linear state estimation techniques (e.g., particle filtering [35]), which are less computationally expensive.

In track-before-detect multi-view tracking, the common view can be based on the foreground likelihood (Equation 10.8) or the binary mask (Equation 10.6, Equation 10.4). The single-target multi-view track-before-detect particle filter can be modified for multiple targets by incorporating particle clustering [22]. The cluster information allows normalizing weights per target/cluster, thus facilitating tracking weak and new born targets.

Figure 10.11 shows a comparison between the evolution of particle weights with and without the cluster-based update strategy. It can be seen that without the multi-target update strategy (Figure 10.11(a)), a target is lost while another has a very low weight that will cause that target to be lost in the subsequent frame.

The particles can be clustered using K-means, Mixture of Gaussians or Mean-shift (MS) [36]. If the total number of targets is not known, a nonparametric clustering technique that does not require prior knowledge of the number of clusters, such as MS, can be used. MS climbs the gradient of a probability distribution to find the nearest dominant mode or peak and does not impose constraints on the shape of the clusters.

An example of tracking results obtained with the multi-target particle filtering track-before-detect (MT-PF-TBD) on the ISSIA<sup>2</sup> dataset is shown in Figure 10.12(c,e). The bandwidth chosen for MS is  $h = 5$ , which is appropriate for clustering particles generated around a target that is affected by blurring; 3000 particles per target are used. It can be seen that most targets are tracked over the entire scene, the exception being the goalkeeper on the left corner of the field. This target is not tracked initially (Figure 10.12(c)) despite being represented with significant information (Figure 10.12(b)) as he was static and hence not following the expected motion model. The prediction resulted in moving all particles away from the target. The corresponding track is generated when he starts moving during the attack on the goal (Figure 10.12(d-e)).

## 10.6 Conclusions

This Chapter discussed and classified techniques for tracking in multiple cameras with partially overlapping fields of view. The Chapter covers the two major groups of multi-view tracking algorithms, namely track-first and fuse-first approaches. Track-first approaches employ tracking in each view as well as on the common view. Trackers in each view can also collaborate with each other to improve the target estimates. Contrary to track-first methods, fuse-first approaches defer tracking until the fusion of target localization information on the common view. Tracking is then performed only once on the common view using multiple single-target trackers or multi-target trackers. Tracking on the common view can be based on detections (when targets are first localized prior to tracking) or on simultaneous detection and tracking. In this context, the Chapter has presented a track-before-detect multi-target particle filter tracker where only pixels following a certain dynamic model are tracked, without any explicit detection mechanisms. This approach not only eliminates the detection step after data fusion, but also helps reducing false positives due to noise.

## References

1. Taj, M., Cavallaro, A.: Audio-assisted trajectory estimation in non-overlapping multi-camera networks. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Taipei, Taiwan (2009)

---

<sup>2</sup> Raw videos courtesy of Institute of Intelligent Systems for Automation - C.N.R., Bari, IT. <http://www.issia.cnr.it>, last accessed: 26 June, 2008

2. Nettleton, E., Durrant-Whyte, H., Sukkarieh, S.: A robust architecture for decentralised data fusion. In: Proceedings of the International Conference on Advanced Robotics, Coimbra, PT (2003)
3. Medeiros, H., Park, J., Kak, A.C.: Distributed object tracking using a cluster-based kalman filter in wireless camera networks. *Journal of Selected Topics In Signal Processing* (2008)
4. Soto, C., Song, B., Roy-Chowdhury, A.: Distributed multi-target tracking in a self-configuring camera network. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 1486–1493 (2009)
5. Kim, K., Davis, L.S.: Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) *ECCV 2006. LNCS*, vol. 3953, pp. 98–109. Springer, Heidelberg (2006)
6. Javed, O., Shafique, K., Shah, M.: Appearance modeling for tracking in multiple non-overlapping cameras. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 26–33 (2005)
7. Morariu, V., Camps, O.: Modeling correspondences for multi-camera tracking using non-linear manifold learning and target dynamics. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2006)
8. Morioka, K., Mao, X., Hashimoto, H.: Global color model based object matching in the multi-camera environment. In: Proceedings of the International Conference on Intelligent Robots and Systems, pp. 2644–2649 (2006)
9. Cai, Q., Aggarwal, J.: Tracking human motion in structured environments using a distributed-camera system. *Transaction on Pattern Analysis and Machine Intelligence* 21, 1241–1247 (1999)
10. Black, J., Ellis, T., Rosin, P.: Multi view image surveillance and tracking. In: Proceedings of the International Workshop on Motion and Video Computing (2002)
11. Anjum, N., Cavallaro, A.: Trajectory association and fusion across partially overlapping cameras. In: Proceedings of the International Conference on Advanced Video and Signal Based Surveillance (2009)
12. Du, W., Piater, J.: Multi-camera people tracking by collaborative particle filters and principal axis-based integration. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) *ACCV 2007. Part I. LNCS*, vol. 4843, pp. 365–374. Springer, Heidelberg (2007)
13. Pham, N., Huang, W., Ong, S.: Probability hypothesis density approach for multi-camera multi-object tracking. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) *ACCV 2007. Part I. LNCS*, vol. 4843, pp. 875–884. Springer, Heidelberg (2007)
14. Qu, W., Schonfeld, D., Mohamed, M.: Distributed bayesian multiple-target tracking in crowded environments using multiple collaborative cameras. *EURASIP Journal on Applied Signal Processing*, 21 (2007)
15. Kayumbi, G., Cavallaro, A.: Multiview trajectory mapping using homography with lens distortion correction. *EURASIP Journal on Image and Video Processing* (2008)
16. Khan, S., Shah, M.: Tracking multiple occluding people by localizing on multiple scene planes. *Trans. on Pattern Analysis and Machine Intelligence* 31, 505–519 (2009)
17. Stauffer, C., Tieu, K.: Automated multi-camera planar tracking correspondence modeling. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2003)
18. del Rincon, J.M., Herrero-Jaraba, J.E., Gmez, J.R., Orrite-Urunuela, C.: Automatic left luggage detection and tracking using multi-camera ukf. In: Proceedings of the International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pp. 51–58 (2006)
19. Focken, D., Stiefelhagen, R.: Towards vision-based 3-d people tracking in a smart room. In: Proceedings of the International Conference on Multimodal Interfaces (2002)

20. Mittal, A., Davis, L.S.: M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) *ECCV 2002*. LNCS, vol. 2350, pp. 18–33. Springer, Heidelberg (2002)
21. Eshel, R., Moses, Y.: Homography based multiple camera detection and tracking of people in a dense crowd. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2008)
22. Taj, M., Cavallaro, A.: Multi-camera track-before-detect. In: Proceedings of the International Conference on Distributed Smart Cameras (2009)
23. Delannay, D., Danhier, N., Vleeschouwer, C.D.: Detection and recognition of sports (wo)man from multiple views. In: Proceedings of the International Conference on Distributed Smart Cameras (2009)
24. Khan, S.M., Shah, M.: A multiview approach to tracking people in crowded scenes using a planar homography constraint. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) *ECCV 2006*. LNCS, vol. 3954, pp. 133–146. Springer, Heidelberg (2006)
25. Yang, D., Gonzalez-Banos, H., Guibas, L.: Counting people in crowds with a real-time network of simple image sensors. In: Proceedings of the International Conference on Computer Vision, pp. 122–129 (2003)
26. Monier, E., Wilhelm, P., Rckert, U.: Multi camera based tracking of indoor team. In: Proceedings of the International Conference on Distributed Smart Cameras (2009)
27. Khan, S., Shah, M.: Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. Transaction on Pattern Analysis and Machine Intelligence 25, 1355–1360 (2003)
28. Fleuret, F., Berclaz, J., Lengagne, R., Fua, P.: Multicamera people tracking with a probabilistic occupancy map. Transaction on Pattern Analysis and Machine Intelligence, 30, 267–282 (2008)
29. Alahi, A., Boursier, Y., Jacquesy, L., Vandergheynst, P.: Sport players detection and tracking with a mixed network of planar and omnidirectional cameras. In: Proceedings of the International Conference on Distributed Smart Cameras (2009)
30. Black, J., Ellis, T.: Multi camera image tracking. Elsevier Journal of Image and Vision Computing 24, 1256–1267 (2006)
31. Jain, A., Flynn, M.M.P.: Data clustering: A review. ACM Computing Surveys 31, 264–323 (1999)
32. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. Transaction on Pattern Analysis and Machine Intelligence 25, 564–577 (2003)
33. Czyz, J., Ristic, B., Macq, B.: A particle filter for joint detection and tracking of color objects. Elsevier Journal of Image and Vision Computing 25, 1271–1281 (2006)
34. Bruno, M.G.S., Moura, J.M.F.: Multiframe detector/tracker: optimal performance. Transaction on Aerospace and Electronic Systems 37, 925–945 (2001)
35. Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking. Transaction on Signal Processing 50, 174–188 (2002)
36. Comaniciu, D., Meer, P.: Distribution free decomposition of multivariate data. Transaction on Pattern Analysis and Machine Intelligence 2, 22–30 (1999)

# Chapter 11

## Shape from Photographs: A Multi-view Stereo Pipeline

Carlos Hernández and George Vogiatzis

**Abstract.** Acquiring 3D shape from images is a classic problem in Computer Vision occupying researchers for at least 20 years. Only recently however have these ideas matured enough to provide highly accurate results. We present a complete algorithm to reconstruct 3D objects from images using the stereo correspondence cue. The technique can be described as a pipeline of four basic building blocks: camera calibration, image segmentation, photo-consistency estimation from images, and surface extraction from photo-consistency. In this Chapter we will put more emphasis on the latter two: namely how to extract geometric information from a set of photographs without explicit camera visibility, and how to combine different geometry estimates in an optimal way.

### 11.1 Introduction

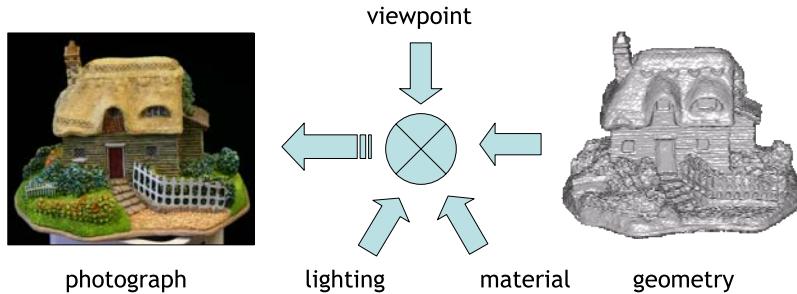
Digital modeling of 3D scenes is becoming increasingly popular and necessary for a wide range of applications such as cultural heritage preservation, online shopping or computer games. Although active methods [34][49] remain one of the most popular techniques of acquiring shape, the high cost of the equipment, complexity, and difficulties to capture color are three big disadvantages. As opposed to active techniques, photograph-based techniques provide an efficient and easy way to acquire shape and color by simply capturing a sequence of photographs of the object.

The goal of any shape-from-photographs algorithm can be described as “*given a set of input photographs, how to estimate a 3D shape that would generate the same photographs, assuming same material, viewpoints and lighting conditions*”. This

---

Carlos Hernández  
Toshiba Research Cambridge, UK  
e-mail: [carlos.hernandez@crl.toshiba.co.uk](mailto:carlos.hernandez@crl.toshiba.co.uk)

George Vogiatzis  
Aston University, Birmingham, UK  
e-mail: [g.vogiatzis@aston.ac.uk](mailto:g.vogiatzis@aston.ac.uk)



**Fig. 11.1** Image formation model. The image of a 3D scene depends on its geometry, material properties, lighting conditions and pose of the viewer.

definition highlights the main difficulty of the problem: photographs are obtained as a result of complex interactions between the geometry of the scene, the materials of the scene, the lighting conditions and the viewpoints (see Figure 11.1). Hence recovering the geometry just from photographs is not only a challenging problem but also, in the general case, an ill-posed problem. It is challenging because lighting and material properties play a very important role in the image formation model. The same geometry with a different material or different lighting conditions can give extremely different photographs. It is also an ill-posed problem because, in the general case, different combinations of geometry, lighting and material can produce exactly the same photographs, making it impossible to recover a single scene geometry. The main recipe to make the problem well-posed is to use priors on the types of surface that one expects. Traditionally the most common type of prior is the smooth surface prior. However when dealing with special classes of objects such as human faces or man-made objects, more evolved priors have been successfully used (*e.g.*, human faces [54], buildings [53] or planes [15]).

As for the importance of materials and lighting conditions, it has been addressed by restricting the class of materials a particular algorithm is designed for. As a result, no single method is able to correctly reconstruct a general scene with any type of materials and lighting conditions, leading to a plethora of specific algorithms designed for specific types of objects and using specific cues: silhouettes [1], texture [50], transparency [44], defocus [14], shading [51] or correspondence, both sparse [3] and dense [40]. Historically the most successful cues have been silhouettes, correspondence, and shading. Silhouettes and correspondences are the most robust of all due to their invariance to illumination changes. The shading cue needs a more controlled illumination environment, but it can produce breathtaking results, which makes it widely used too. An example of an algorithm [23] exploiting the shading cue is shown in Figure 11.2. The algorithm is designed to find a 3D shape that produces the same shading as the original object. Interestingly, if the estimated 3D shape is then used to manufacture a replica from a different material (in Figure 11.2 the original is porcelain, while the replica is plaster) we can appreciate how the

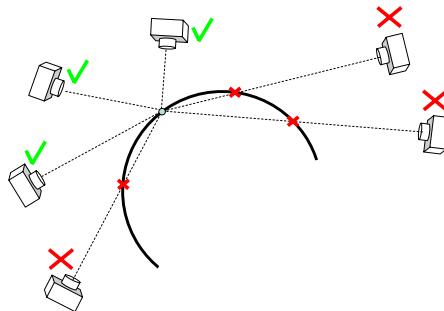


**Fig. 11.2** Shading comparison of a porcelain figurine and a manufactured replica obtained using [23]. The original porcelain figurine is shown on the left, while a manufactured replica using the 3D model obtained using [23] is shown on the right. The material of the replica is plaster. See how the replica perfectly imitates the shading component, even though the materials are different.

replica still shows the same shading pattern. This is the desired behavior, since the algorithm is specifically designed to imitate the shading, not to produce identical photographs.

Among the vast literature available on image-based modeling techniques, recent work on multi-view stereo (MVS) reconstruction has become a growing area of interest in recent years with many differing techniques achieving a high degree of accuracy [40]. These techniques are mainly based on the correspondence cue and focus on producing 3D models from a sequence of calibrated images of an object, where the intrinsic parameters and pose of the camera are known. In addition to providing a taxonomy of methods, [40] also provides a quantitative analysis of performance both in terms of accuracy and completeness. If we take a look at the top performers, they may be loosely divided into two groups. The first group makes use of techniques such as correspondence estimation, local region growing and filtering to build up a “*cloud of patches*” [17, 19, 35, 36] that can be optionally made dense using meshing algorithms such as Poisson reconstruction [4] or signed distance functions [12]. The second group makes use of some form of global optimization strategy on a volumetric representation to extract a surface [18, 20, 24, 47, 48]. Under this second paradigm, a 3D cost volume is computed, and then a 3D surface is extracted using tools previously developed for the 3D segmentation problem such as deformable models [20], level-sets [13, 39] or graph-cuts [6, 16, 24, 33, 41, 46, 48].

The way volumetric methods usually exploit photo-consistency is by building a 3D map of photo-consistency where each 3D location gives an estimate of how photo-consistent would be the reconstructed surface at that location. The only requirement to compute this photo-consistency 3D map is that camera visibility is



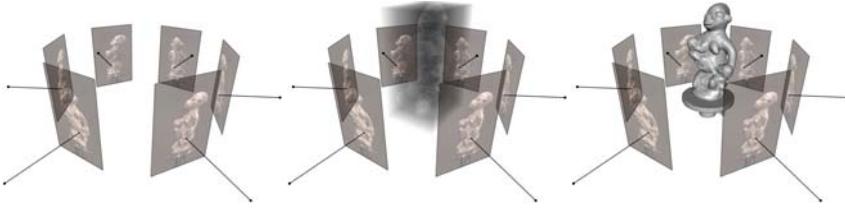
**Fig. 11.3** Occlusion problem. In order to compute shape using photo-consistency, the camera visibility is required. At the same time, in order to compute the camera visibility, the shape is required.

available. Unfortunately, the geometry of the scene, *i.e.*, what we try to compute, is required to know which cameras see a 3D location (see Figure 11.3). In order to break this dependency between visibility and shape, multi-view stereo algorithms have taken different approaches. A majority of methods use the notion of “*current surface*” in order to jointly optimize for camera visibility and shape. The visibility computed from the reconstructed surface at iteration  $i - 1$  is then used to compute photo-consistency at iteration  $i$ , improving the reconstruction gradually [13]. Some methods use a proxy of the true surface to estimate visibility, such as the visual hull [24, 48]. Finally, a third category of methods tries to compute a “*visibility-independent*” photo-consistency where occlusion is treated as an additional source of image noise [7, 18, 20].

In this Chapter we will give further insight into a two-stage MVS volumetric approach: namely how to extract a 3D volume of photo-consistency from a set of photographs without explicit camera visibility in Section 11.3, and how to extract a surface from the photo-consistency volume in a globally optimal way in Section 11.4. The pipeline described in this Chapter is currently a top performer in the recent evaluation of multi-view stereo algorithms by Seitz et al [40].

## 11.2 Multi-view Stereo Pipeline: From Photographs to 3D Models

There exists a vast literature on multi-view stereo algorithms. Even though many of the methods share the same basic architecture, they differ mainly in what type of scenes or computation time they are optimized to work with. All the multi-view stereo methods use the correspondence cue, which is usually exploited in the form of a photo-consistency metric such as Normalized Cross Correlation, Sum of Square Differences, or Mutual Information. Starting from the photo-consistency metric, different algorithms focus on different target applications such as outdoor scenes [45],



**Fig. 11.4** 3D multi-view stereo pipeline. Image calibration, photo-consistency 3D map from a set of photographs (Section 11.3) and surface extraction from a photo-consistency 3D map (Section 11.4).

building reconstruction [11, 37, 38], interior buildings [15] or object reconstruction [40]. In this Chapter we describe a volumetric multi-view stereo approach that is optimized for general scene reconstruction, with a preference for watertight surfaces. The pipeline (see Figure 11.4) can be described as:

- photograph acquisition,
- camera calibration,
- computing 3D photo-consistency from a set of calibrated photographs,
- extracting a 3D surface from a 3D map of photo-consistency.

In the following Sections we focus on how to extract 3D photo-consistency from a set of photographs (see Section 11.3) and how to use the 3D photo-consistency to extract a 3D surface (see Section 11.4). We leave the discussion on image acquisition, *e.g.*, real-time vs photograph-based, and on camera calibration for future discussion (see [43] for an state-of-the-art system to calibrate a set of photographs).

### 11.3 Computing Photo-Consistency from a Set of Calibrated Photographs

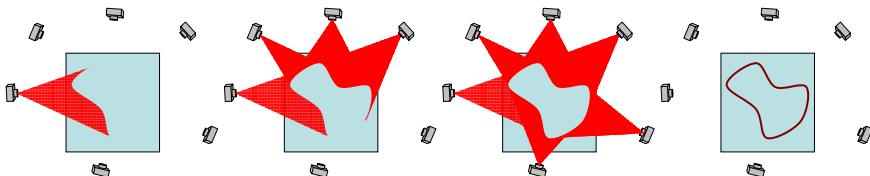
Given a set of images and their corresponding camera poses, we would like to extract a 3D map of photo-consistency that tell us how photo-consistent is a particular 3D location **for a given set of visible cameras**. The main difficulty of this step is how to produce a volumetric measure of photo-consistency without the knowledge of the set of cameras that should be used to compute photo-consistency for every 3D location.

This problem is addressed in the proposed 3D modeling pipeline by following a similar approach to [20] where photo-consistency is made robust to occlusion. This approach computes a 3D map of photo-consistency as an aggregation of depth-maps from different view-points (see Figure 11.5). The creation of such a photo-consistency 3D map is similar in spirit to the space carving approach proposed by [32]. However, by computing it as an aggregation of depth-maps, two advantages appear:

- depth-map computation using dense stereo is a very successful and active research topic. It is an ideal building block to use since improvements in the field of dense stereo can be directly beneficial to the multi-view stereo problem.
- Computation time is no longer dependent on the resolution of the 3D volume, but on the number of cameras. It is also highly parallelizable, since each depth-map is independently computed and no iterated visibility computation is required.

By building a 3D map of photo-consistency, the 3D reconstruction problem can now be seen as a 3D segmentation task, allowing us to use algorithms previously developed for 3D segmentation. These algorithms include deformable surfaces [20], Poisson reconstruction [17], signed distance functions [18], Delaunay [7] or MRFs [22, 29, 47].

A comparison of the importance of this stage in the reconstruction pipeline is shown in Figure 11.6. The occlusion-robust photo-consistency of [20] (Figure 11.6 middle) clearly outperforms [48] (Figure 11.6 left). However, since this method exploit the redundancy between images to be robust against occlusion, it suffers with sparse data sets (see the missing vertical wall in Figure 11.6 middle). An improved version of the occlusion-robust photo-consistency has been proposed in [8] that is capable of better dealing with sparse data sets (see the improvement in the vertical wall in Figure 11.6 right). We adopt [8] in our multi-view stereo pipeline as the



**Fig. 11.5** Computing a photo-consistency volume as aggregation of depth-maps. From left to right, three different stages of merging individual depth-maps into a single photo-consistency volume. Right shows the final photo-consistency volume.



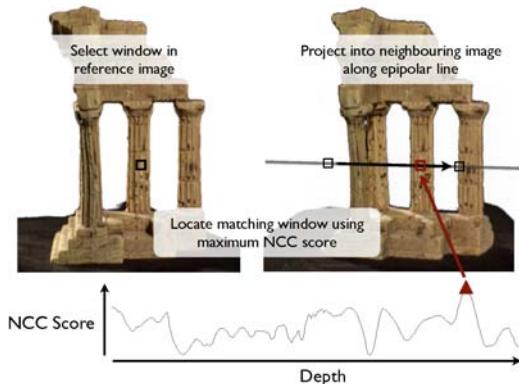
**Fig. 11.6** Noise reduction in photo-consistency. Left: a slice of the photo-consistency used in [48] contains falsely photo-consistent regions (e.g., near the corners). Middle: occlusion robust photo-consistency proposed in [20] significantly suppresses noise and the correct surface can be accurately localized. One side of the vertical wall is missing due to heavy occlusions. Right: occlusion robust photo-consistency proposed in [8]. The vertical wall is correctly represented.

building block to compute individual depth-maps. In the remaining of this Section we describe this algorithm more in detail.

### 11.3.1 Normalized Cross Correlation for Depth-Map Computation

Normalized Cross Correlation (NCC) may be used to define an error metric for matching two windows in different images. Figure 11.7 provides an example of using NCC and epipolar geometry to perform window based matching. If we fix a pixel location in a reference image, for each possible depth away from that pixel we get a corresponding pixel in the second image. By computing the NCC between windows centered in those two pixels we can define a matching score as a function of depth for the reference pixel. We refer to this function as the *correlation curve* of the pixel. A typical correlation curve will exhibit a very sharp peak at the correct depth, and possibly a number of secondary peaks in other depths.

In [20] a depth-map is generated for each input image using this matching technique for neighboring images. For each pixel a number of correlation curves are computed (using a few of the neighboring viewpoints) and the depth that gives rise to most peaks in those curves is selected as the depth for that pixel. See [20] or [47] for details. This process results in an independent depth estimate for each pixel. These depth estimates will unavoidably contain a significant percentage of outliers which must be dealt with in the subsequent step of [20] which is the volumetric fusion of multiple depth-maps. In data sets with a large number of images this is overcome by the redundancy in the depth-estimates. The same surface point is expected to be covered by many different depth-maps, some of which will have the right depth estimate. In sparse data-sets however, each surface point may be seen by as few as two or three depth-maps. It is therefore crucial that outliers are minimized in the depth-map generation stage.



**Fig. 11.7** Normalized Cross-Correlation based window matching.

In order to efficiently exploit NCC as a photo-consistency measure, we need to focus on the two most significant failure modes of NCC matching which are (1) the presence of repetitions in the texture and (2) complete matching failure due to occlusion, distortion and lack of texture. These are now described in more detail.

### 11.3.1.1 Repeating Texture

In general, there is no guarantee that the appearance of a patch is unique across the surface of the object. This results in correlation curve peaks at incorrect depths due to repeated texture — ‘false’ matches (Figure 11.7). A larger window size is more likely to uniquely match to the true surface, reducing the number of false matches. However the associated peak will be broader and less well localized, reducing the accuracy of the depth estimate. The absolute value of the NCC score at a peak reflects how well the two windows match. Thus one might expect the peak with the maximum score to be the true peak. Unfortunately, the appearance of false matches due to repeated texture may result in false peaks having similar or even greater scores than the true surface peak (Figure 11.8 (a)). To identify the correct peak, we propose to apply a spatial consistency constraint across neighboring pixels in the depth-map. The underlying assumption is that if a peak corresponds to the true surface, the neighboring pixels should have peaks at a similar depth. The exception to this is occlusion boundaries, which are however catered for under the next failure mode.

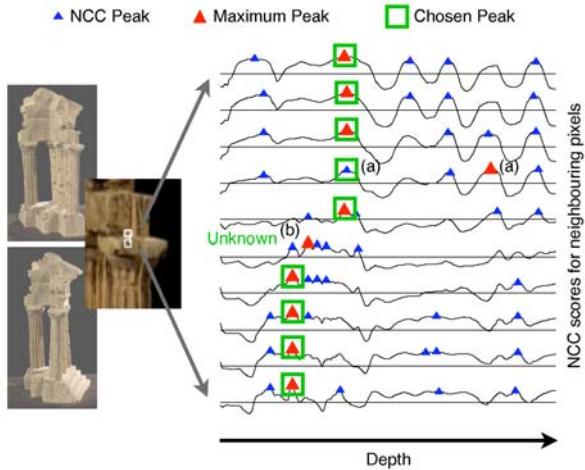
### 11.3.1.2 Matching Failure

The second failure mode is comprised of occlusion errors, distorted image windows (due to slanted surfaces) and lack of texture. In all of these cases, the correlation curve will not exhibit a peak at the true depth of the surface, resulting in only false peaks. Furthermore no spatial consistency can be enforced between the pixel in question and its neighbors. In this situation we would like to acknowledge that the depth at this pixel is unknown and should therefore offer no vote for the surface location.

In order to achieve these two goals we propose an optimization strategy which makes use of a discrete label Markov Random Field (MRF). The MRF allows each pixel to choose a depth corresponding to one of the top NCC peaks which is spatially consistent with neighboring pixels or select an *unknown* label to indicate that no such peak occurs and there is no correct depth estimate. This process means that the returned depth map should only contain accurate depths, estimated with a high degree of certainty, and an *unknown* label for pixels which have no certain associated depth. Figure 11.8 illustrates the optimization for a 1D example of neighboring pixels across an occlusion boundary.

## 11.3.2 Depth Map Estimation

The proposed algorithm estimates the depth for each pixel in the input images. It proceeds in two stages: Initially we extract a set of possible depth values for each pixel using NCC as a matching metric. We then solve a multi-label discrete MRF



**Fig. 11.8** Illustration of the MRF optimization applied to neighboring pixels. Existing method return the maximum peak which results in outliers in the depth estimate. The MRF optimization corrects an outlier to the true surface peak (a) and introduces an unknown label at the occlusion boundary (b)

model which yields the depth assignment for every pixel. One of the key features in this process is the inclusion of an *unknown* state in the MRF model. This state is selected when there is insufficient evidence for the correct depth to be found.

### 11.3.2.1 Candidate Depths

The input to the algorithm is a set of calibrated images  $\mathcal{I}$  and the output is a set of corresponding depth-maps  $\mathcal{D}$ . In the following, we describe how to acquire a depth-map for a reference image  $I_{\text{ref}} \in \mathcal{I}$ . Let  $N(I_{\text{ref}})$  denote a set of ‘neighboring’ images to  $I_{\text{ref}}$ .

As proposed in Section 11.3.1, we wish to obtain a hypothesis set of possible depths for each pixel  $p_i \in I_{\text{ref}}$ . Taking each pixel in turn, we project the epipolar ray into a second image  $I_n \in I_{\text{ref}}$  and sample the NCC matching score over a depth range  $\rho_i(z)$ . We compute the score using a rectangular window centered at the projected image co-ordinates. One of the advantages of the multiple depth hypotheses is the ability to use a smaller matching window to provide a faster computation and improved localization of the surface. Once we have obtained the sampled ray we store the top  $K$  peaks  $\hat{\rho}_i(z_{i,k}), k \in [1, K]$  with the greatest NCC score for each pixel. Depending on the number of images available, and the width of the camera baseline, this process may be repeated for other neighboring images. We then continue to the optimization stage with a set of the best  $K$  possible depths, and their corresponding NCC scores, over all neighboring images of  $I_{\text{ref}}$ .

### 11.3.2.2 MRF Formulation

At this stage a set of candidate depths  $\hat{\rho}_i(z_{i,k}), k \in [1, K]$ , for each pixel  $p_i$  in the reference image  $I_{\text{ref}}$  has been assigned and we wish to determine the correct depth map label for each pixel. As described in Section 11.3.1, we also make use of an *unknown* state to account for the failure modes of NCC matching.

We model the problem as a discrete MRF where each pixel has a set of up to  $(K + 1)$  labels. The first  $K$  labels, fewer if an insufficient number of peaks were found during the matching stage, correspond to the peaks in the NCC function and have associated depths  $z_{i,k} \in \mathcal{Z}_i$  and scores  $\hat{\rho}_i(z_{i,k})$ . The final state is the *unknown* state  $\mathcal{U}$ . If the optimization returns this state, the pixel is not assigned a depth in the final depth map. For each pixel we therefore form an augmented label set  $z'_{i,k} \in \{\mathcal{Z}_i, \mathcal{U}\}$  to include the unknown state.

The optimization assigns a label  $\bar{k}_i \in \{1 \dots K, \mathcal{U}\}$  to each pixel  $p_i$ . The cost function to be minimized consists of unary potentials for each pixel and pairwise interactions over first order cliques. The cost of a labeling  $\bar{\mathbf{k}} = \{\bar{k}_i\}$  is expressed as

$$E(\bar{\mathbf{k}}) = \sum_i \phi(\bar{k}_i) + \sum_{(i,j)} \psi(\bar{k}_i, \bar{k}_j) \quad (11.1)$$

where  $i$  denotes a pixel and  $(i, j)$  denote neighboring pixels.

The following Sections discuss the formulation of the unary potentials  $\phi(\cdot)$  and pairwise interactions  $\psi(\cdot, \cdot)$ .

### 11.3.2.3 Unary Potentials

The unary labeling cost is derived from the NCC score of the peak. We wish to penalize peaks with a lower matching score since they are more likely to correspond to an incorrect match due to occlusion or noise. The NCC process will always return a score in the range  $[-1, 1]$ . As is common practice, [47], we take an inverse exponential function to map this score to a positive cost.

The unary cost for the *unknown* state is set to a constant value  $\phi_U$ . This term serves two purposes. Firstly it acts as a cut-off threshold for peaks with poor NCC scores which have no pairwise support (neighboring peaks of similar depth). This mostly accounts for peaks which are weakly matched due to distortion or noise. Secondly it acts as a truncation on the depth disparity cost of the pairwise term. By assigning a low pairwise cost between peaks and the *unknown* state, the constant unary cost will effectively act as a threshold on the depth disparity to handle the case of an occlusion boundary. Thus the final unary term is given by

$$\phi(k_i = x) = \begin{cases} \lambda e^{-\beta \hat{\rho}_i(z_{i,x})} & x \in [1 \dots K] \\ \phi_U & x = \mathcal{U} \end{cases}. \quad (11.2)$$

### 11.3.2.4 Pairwise Interactions

The pairwise labeling cost is derived from the disparity in depths of neighboring peaks. As has been previously mentioned, this term is not intended to provide a strong regularization of the depth map. Instead it is used to try and determine the correct peak, corresponding to the true surface location, out of the returned peaks. We observe that the correct peak may not have the maximum score. Therefore if there is strong agreement on depth between neighboring peaks, we take this to be the true location of the surface.

When dealing with the depth disparity term we are really considering surface orientation; whether the surface normal is pointing towards or away from the camera. Under a perspective projection camera model it is therefore necessary to correct for the absolute depth of the peaks rather than simply taking the difference in depth. We perform this correction by dividing by the average depth of the two peaks. The resulting pairwise term is given by

$$\psi(k_i = x, k_j = y) = \begin{cases} 2 \frac{|z_{i,x} - z_{j,y}|}{(z_{i,x} + z_{j,y})} & x \in [1 \dots K] \quad y \in [1 \dots K] \\ \psi_U & x = \mathcal{U} \quad y \in [1 \dots K] \\ \psi_U & x \in [1 \dots K] \quad y = \mathcal{U} \\ 0 & x = \mathcal{U} \quad y = \mathcal{U} \end{cases}. \quad (11.3)$$

We set  $\psi_U$  to a small value to encourage regions with many pixels labeled as *unknown* to coalesce. This acts as a further stage of noise reduction since it prevents spurious peaks with high scores but no surrounding support from appearing in regions of occlusion.

### 11.3.2.5 Optimization

To obtain the final depth map we need to determine the optimal labeling  $\hat{\mathbf{k}}$  such that

$$E(\hat{\mathbf{k}}) = \arg \min_{(\bar{\mathbf{k}})} \sum_i \phi(\bar{k}_i) + \sum_{(i,j)} \psi(\bar{k}_i, \bar{k}_j). \quad (11.4)$$

Since in the general case this is an NP-hard problem we must use an approximate minimization algorithm to achieve a solution. The most well-known techniques for solving problems of this nature are based on graph-cuts and belief propagation. Instead, we use the recently developed sequential tree-reweighted message passing algorithm, termed TRW-S, of [30]. This has been shown to outperform belief propagation and graph-cuts in tests on stereo matching using a discrete number of disparity levels. In addition to minimizing the energy, the algorithm estimates a lower bound on the energy at each iteration which is useful in checking for convergence

and evaluating the performance of the algorithm. We should note, however, that we are by no means guaranteed that the lower bound is attainable.

### 11.3.3 Photo-Consistency 3D Map from a Set of Depth-Maps

In order to create a 3D volume of photo-consistency from a set of depth-maps  $\mathcal{D}$  we “*uplift*” every depth-map in  $\mathcal{D}$  into 3D using the camera calibration data. The photo-consistency of a 3D point  $\mathbf{x}$  is defined as the sum of the confidences of all its nearby depth-map points. That is, given all the uplifted depth-map 3D points  $\mathbf{d}_i$  and their corresponding confidence values  $s_i$ , the photo-consistency  $\mathcal{C}(\mathbf{x})$  can be defined as

$$\mathcal{C}(\mathbf{x}) = \sum_{i:|\mathbf{x}-\mathbf{d}_i|<\varepsilon} s_i, \quad (11.5)$$

where  $\varepsilon$  is a pre-defined ball size. If the photo-consistency is to be discretized using a volumetric grid, then  $\varepsilon$  is simply the size of a voxel.

## 11.4 Extracting a 3D Surface from a 3D Map of Photo-Consistency

Given a 3D map of photo-consistency, we would like to extract a 3D surface. As mentioned earlier, by building a 3D map of photo-consistency, the reconstruction problem can now be solved using 3D segmentation techniques. Out of all the segmentation algorithms available, MRF approaches are very widely spread due to its global convergence properties. They also allow the fusion of different cues in an elegant way (*e.g.*, see [29]). One of the main criticisms of MRFs applied to 3D segmentation is the discretization artifacts originating from its discrete nature. In order to remove them, the surface is usually further refined using a continuous formulation such as level-sets [13,39] or deformable models [20], allowing for a finer control of the regularization than the one provided by MRFs. In the remaining of this Section we describe the MRF framework for multi-view stereo first proposed by [47] and further extended in [22]. We also describe the deformable model by [20] that we use as a refinement step.

### 11.4.1 Multi-view Stereo Using Multi-resolution Graph-Cuts

In [5] and subsequently in [2] it was shown how graph-cuts can optimally partition 2D or 3D space into ‘foreground’ and ‘background’ regions under any cost functional consisting of the following two terms:

- **Labeling cost or unary term:** for every point in space there is a cost for it being labeled ‘foreground’ or ‘background’.
- **Discontinuity cost or binary term:** for every point in space, there is a cost for it lying on the boundary between the two partitions.

Mathematically, the cost functional described above can be seen as the sum of a weighted *surface area* of the boundary surface and a weighted *volume* of the ‘foreground’ region as follows:

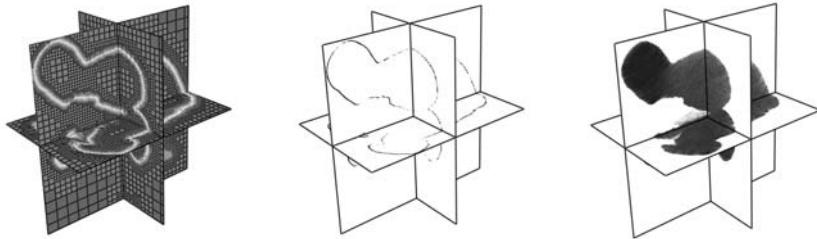
$$E(S) = \int_S \rho(\mathbf{x}) dA + \int_{V(S)} \sigma(\mathbf{x}) dV \quad (11.6)$$

where  $S$  is the boundary between ‘foreground’ and ‘background’,  $V(S)$  denotes the ‘foreground’ volume enclosed by  $S$  and  $\rho$  and  $\sigma$  are two scalar density fields. The application described in [5] was the problem of 2D/3D segmentation. In that domain  $\rho(\mathbf{x})$  is defined as a function of the image intensity gradient and  $\sigma(\mathbf{x})$  as a function of the image intensity itself or local image statistics.

In the framework of the multi-view stereo problem, this model balances two competing terms: the first one minimizes a surface integral of photo-consistency (binary term) while the second one maximizes the volume of regions with a high evidence of being foreground (unary term). In the literature, it is usually the binary term that is data driven, while the unary term is just used as a basic prior, *e.g.*, as a ballooning term [9]. In this work, we use the photo-consistency 3D map computed in Section 11.3 as the binary term. As for the unary term, very little work has been done to obtain an appropriate ballooning term. In most of the previous work on volumetric multi-view stereo the ballooning term is a very simplistic inflationary force that is constant in the entire volume, *i.e.*,  $\sigma(\mathbf{x}) = -\lambda$ . This simple model tries to recover thin structures by maximizing the volume inside the final surface. However, as a side effect, it also fills in concavities behaving as a regularization force and smoothing fine details.

When silhouettes of the object are available, an additional *silhouette cue* can be used [24, 48], which provides the constraint that all points *inside* the object volume must project inside the silhouettes of the object. Hence the silhouette cue can provide some foreground/background information by giving a very high likelihood of being *outside* the object to 3D points that project outside the silhouettes. However this ballooning term is not enough if thin structures or big concavities are present, in which case the method fails (see Figure 11.16 middle row). Very recently, a data driven, foreground/background model based on the concept of *photo-flux* has been introduced [6]. However, the approach requires approximate knowledge of the object surface orientation which in many cases is not readily available.

Ideally, the ballooning term should be linked to the notion of visibility, where points that are not visible from any camera are considered to be inside the object or **foreground**, and points that are visible from at least one camera are considered to be outside the object or **background**. An intuition of how to obtain such a ballooning term is found in a classic paper on depth sensor fusion by Curless and Levoy [12]. In that paper the authors fuse a set of depth sensors using signed distance functions. This fusion relies on the basic principle that the space between the sensor and the depth map should be empty or background, and the space after the depth map should be considered as foreground. In this Section we follow the approach by [22] where this visibility principle is generalized and computed in a probabilistic version by



**Fig. 11.9** Different terms used in the graph-cut algorithm to reconstruct the Gormley sculpture of Figure 11.16. Left: multi-resolution grid used in the graph-cut algorithm. Middle: Discontinuity cost  $\rho(\mathbf{x})$  (or photo-consistency). Right: labeling cost  $\sigma(\mathbf{x})$  (or intelligent ballooning).

calculating the “*evidence of visibility*” from a given set of depth-maps. The “*evidence of visibility*” is then used as an intelligent ballooning term.

The outline of the full system is as follows:

- create a set of depth-maps from the set of input calibrated photographs,
- compute the photo-consistency 3D map from the set of depth-maps,
- derive the discontinuity cost  $\rho(\mathbf{x})$  from the photo-consistency 3D map,
- derive the labeling cost  $\sigma(\mathbf{x})$  from the set of depth-maps, *i.e.*, use a data-aware ballooning term computed from the evidence of visibility and,
- extract the final surface as the global solution of the min-cut problem given  $\rho(\mathbf{x})$  and  $\sigma(\mathbf{x})$ .

A real example of discontinuity and labeling costs is shown in Figure 11.9. Note they have been computed on a multi-resolution grid.

The algorithm just described can also be used when the input is no longer a set of photographs but a set of depth-maps obtained from other types of sensor, *e.g.*, laser scanner. In this case, the system just skips the first step, since the depth-maps are already available, and computes  $\rho$  and  $\sigma$  directly from the set of depth-maps given as input.

### 11.4.2 Discontinuity Cost from a Set of Depth-Maps

Once we have computed a depth-map for every input image, we can build the photo-consistency 3D map ( $\mathbf{x}$ ) for every 3D location  $\mathbf{x}$  as explained in Section 11.3.3. Since the graph-cut algorithm **minimizes** the discontinuity cost, and we want to *maximize* the photo-consistency, we invert the discontinuity map  $\rho(\mathbf{x})$  using the exponential:

$$\rho(\mathbf{x}) = e^{-\mu C(\mathbf{x})}, \quad (11.7)$$

where  $\mu$  is a very stable rate-of-decay parameter that converts photo-consistency scores into a normalized discontinuity cost in the range  $[0, 1]$ .

As a way of improving the big memory requirements of graph-cut methods, we propose to store the values of  $\rho(\mathbf{x})$  in an octree partition of 3D space. The size of the octree voxel will depend on the photo-consistency value  $\mathcal{C}(\mathbf{x})$ . Voxels with a non-zero photo-consistency value will have the finest resolution while the remaining space where  $\mathcal{C}(\mathbf{x}) = 0$  will be partitioned using bigger voxels, the voxel size being directly linked with the distance to the closest non-empty voxel (see Figure 11.9 for an example of such an octree partition).

### 11.4.3 Graph Structure

To obtain a discrete solution to Equation 11.6 3D space is quantized into voxels using an octree partition. The graph nodes consist of all voxels whose centers are within a certain bounding box that is guaranteed to contain the object. For the results presented here these nodes were connected with a regular 6-neighborhood grid. Bigger neighborhood systems can be used which provide a better approximation to the continuous functional 11.6, at the expense of using more memory to store the graph. Now assume two voxels centered at  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are neighbors. Let the smaller voxel be size  $h \times h \times h$ . Then the weight of the edge joining the two corresponding nodes on the graph will be [5]

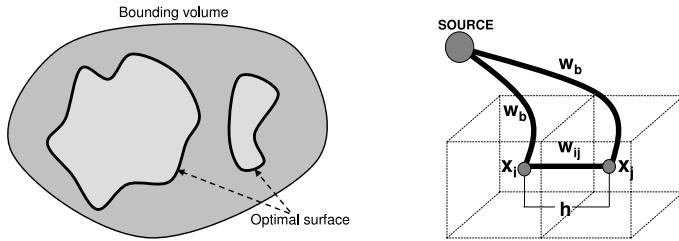
$$w_{ij} = \frac{4\pi h^2}{3} \rho \left( \frac{\mathbf{x}_i + \mathbf{x}_j}{2} \right) \quad (11.8)$$

where  $\rho(\mathbf{x})$  is the matching cost function defined in 11.7. In addition to these weights between neighboring voxels there is also the ballooning force edge connecting every voxel to the source node with a constant weight of  $w_b = \lambda h^3$ . Finally, the outer voxels that are part of the bounding box (or the voxels outside the visual hull if that is available) are connected with the sink with edges of infinite weight. The configuration of the graph is shown in Figure 11.10 (right).

It is worth pointing out that the graph structure described above can be thought of as a simple binary MRF. Variables correspond to voxels and can be labeled as being *inside* or *outside* the scene. The unitary clique potential is just 0 if the voxel is outside and  $w_b$  if it is inside the scene while the pairwise potential between two neighbor voxels  $i$  and  $j$  is equal to  $w_{ij}$  if the voxels have opposite labels and 0 otherwise. As a binary MRF with a *sub-modular* energy function [31] it can be solved exactly in polynomial time using Graph-cuts.

### 11.4.4 Labeling Cost from a Set of Depth-Maps

In the same way as the computation of the discontinuity cost, the ballooning term  $\sigma(\mathbf{x})$  can be computed exclusively from a set of depth-maps. We propose to use the probabilistic evidence for visibility proposed by [22] and described in Section 11.4.5 as an *intelligent* ballooning term. To do so, all we need is to choose a noise model for the sensor given a depth-map  $D$  and its confidence  $\mathcal{C}(D)$ . We propose



**Fig. 11.10** Surface geometry and flow graph construction. On the left: a 2D slice of space showing the bounding volume and the optimal surface inside it that is obtained by computing the minimum cut of a weighted graph. Note that complicated topologies such as holes or disjoint volumes can be represented by the model and recovered after optimization. On the right: the correspondence of voxels with nodes in the graph. Each voxel is connected to its neighbors as well as to the source.

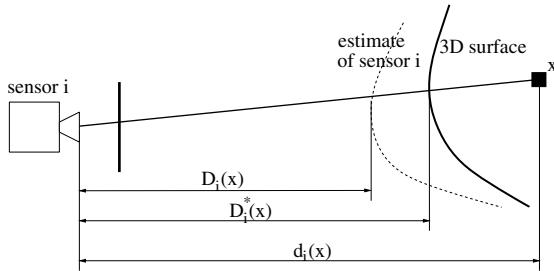
to use a simplistic yet powerful model of a Gaussian contaminated with a uniform distribution, *i.e.*, an inlier model plus an outlier model. The inlier model is assumed to be a Gaussian distribution centered around the true depth. The standard deviation is considered to be a constant value that only depends on the image resolution and camera baseline. The outlier ratio varies according to the confidence on the depth estimation  $\mathcal{C}(D)$ , and in this work is just proportional to it. The labeling cost  $\sigma(\mathbf{x})$  at a given location is just the evidence of visibility. The details of this calculation are laid out in the next Section.

#### 11.4.5 Probabilistic Fusion of Depth Sensors

This Section considers the problem of probabilistically fusing depth maps obtained from  $N$  depth sensors. We will be using the following notation: The sensor data is a set of  $N$  depth maps  $\mathcal{D} = D_1, \dots, D_N$ . A 3D point  $\mathbf{x}$  can therefore be projected to a pixel of the depth map of the  $i$ -th sensor and the corresponding depth measurement at that pixel is written as  $D_i(\mathbf{x})$  while  $D_i^*(\mathbf{x})$  denotes the true depth of the 3D scene. The measurement  $D_i(\mathbf{x})$  contains some noise which is modeled probabilistically by a pdf conditional on the real surface depth

$$p(D_i(\mathbf{x}) | D_i^*(\mathbf{x})). \quad (11.9)$$

The depth of the point  $\mathbf{x}$  away from the sensor is  $d_i(\mathbf{x})$  (see Figure 11.11). If  $\mathbf{x}$  is located on the 3D scene surface then  $\forall i D_i^*(\mathbf{x}) = d_i(\mathbf{x})$ . If for a particular sensor  $i$  we have  $D_i^*(\mathbf{x}) > d_i(\mathbf{x})$  this means that the sensor can *see beyond*  $\mathbf{x}$  or in other words that  $\mathbf{x}$  is *visible* from the sensor. We denote this event by  $V_i(\mathbf{x})$ . When the opposite event  $\bar{V}_i(\mathbf{x})$  is true, as in Figure 11.11, then  $\mathbf{x}$  is said to be *occluded* from the sensor.



**Fig. 11.11** Sensor depth notation. Sensor  $i$  measures the depth of the scene along the optic ray from the sensor to 3D point  $\mathbf{x}$ . The depth of point  $\mathbf{x}$  from sensor  $i$  is  $d_i(\mathbf{x})$  while the correct depth of the scene along that ray is  $D_i^*(\mathbf{x})$  and the sensor measurement is  $D_i(\mathbf{x})$ .

To fuse these measurements we consider a predicate  $V(\mathbf{x})$  which is read as: ‘ $\mathbf{x}$  is visible from at least one sensor’. More formally the predicate is defined as follows:

$$V(\mathbf{x}) \equiv \exists i V_i(\mathbf{x}) \quad (11.10)$$

$V(\mathbf{x})$  acts as a proxy for the predicate we should ideally be examining which is ‘ $\mathbf{x}$  is outside the volume of the 3D scene’. However the sensors cannot provide any evidence beyond  $D_i^*(\mathbf{x})$  along the optic ray, the rest of the points on that ray being occluded. If there are locations that are occluded from all sensors, no algorithm could produce any evidence for these locations being inside or outside the volume. In that sense therefore,  $V(\mathbf{x})$  is the strongest predicate one could hope for in an optical system. An intuitive assumption made throughout this Section is that the probability of  $V(\mathbf{x})$  depends only on the depth measurements of sensors along optic rays that go through  $\mathbf{x}$ . This means that most of the inference Equations will be referring to a single point  $\mathbf{x}$ , in which case the  $\mathbf{x}$  argument can be safely removed from the predicates.

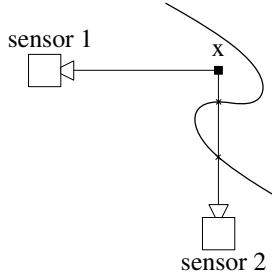
The set of assumptions which we denote by  $\mathcal{J}$  consists of the following:

- The probability distributions of the true depths of the scene  $D_1^*(\mathbf{x}) \cdots D_N^*(\mathbf{x})$  and also of the measurements  $D_1(\mathbf{x}) \cdots D_N(\mathbf{x})$  are independent given  $\mathcal{J}$  (see Figure 11.12 for justification).
- The probability distribution of a sensor measurement given the scene depths and all other measurements only depends on the surface depth it is measuring:

$$p(D_i | D_1^* \cdots D_N^* D_{j \neq i} \mathcal{J}) = p(D_i | D_i^* \mathcal{J}) \quad (11.11)$$

We are interested in computing the evidence function under this set of independence assumptions [26] for the visibility of the point given all the sensor measurements:

$$e(V | D_1 \cdots D_N \mathcal{J}) = \log \frac{p(V | D_1 \cdots D_N \mathcal{J})}{p(\overline{V} | D_1 \cdots D_N \mathcal{J})}. \quad (11.12)$$



**Fig. 11.12** Visibility from sensors. In the example shown above the point is not visible from sensor 2 while it is visible from sensor 1, i.e., we have  $V_1 \bar{V}_2$ . In the absence of a surface prior that does not favor geometries such as the one shown above, one can safely assume that there is no probabilistic dependence between visibility or invisibility from any two sensors.

From  $\mathcal{J}$  and rules of probability one can derive:

$$p(\bar{V} | D_1 \dots D_N \mathcal{J}) = \prod_{i=1}^N p(\bar{V}_i | D_i \mathcal{J}). \quad (11.13)$$

and

$$p(\bar{V}_i | D_i \mathcal{J}) = \frac{\int_0^{d_i} p(D_i | D_i^* \mathcal{J}) p(D_i^* | \mathcal{J}) dD_i^*}{\int_0^\infty p(D_i | D_i^* \mathcal{J}) p(D_i^* | \mathcal{J}) dD_i^*} \quad (11.14)$$

As mentioned, the distributions  $p(D_i | D_i^* \mathcal{J})$  encode our knowledge about the measurement model. Reasonable choices would be the Gaussian distribution or a Gaussian contaminated by an outlier process. Both of these approaches are evaluated in Section 11.5. Another interesting option would be multi-modal distributions. The prior  $p(D_i^* | \mathcal{J})$  encodes some geometric knowledge about the depths in the scene. In all the examples presented a bounding volume was given so we assumed a uniform distribution of  $D_i^*$  inside that volume.

If we write  $\pi_i = p(\bar{V}_i | D_i \mathcal{J})$  then the evidence for visibility is given by:

$$e(V | D_1 \dots D_N \mathcal{J}) = \log \frac{1 - \pi_1 \dots \pi_N}{\pi_1 \dots \pi_N}. \quad (11.15)$$

In the following Section we point out an interesting connection between the probabilistic visibility approach and one of the classic methods in the Computer Graphics literature for merging range data.

#### 11.4.5.1 Signed Distance Functions

In [12], Curless and Levoy compute signed distance functions from each depth-map (positive towards the camera and negative inside the scene) whose weighted average is then stored in a 3D scalar field. So if  $w_i(\mathbf{x})$  represents the confidence of depth measurement  $D_i(\mathbf{x})$  in the  $i$ -th sensor, the 3D scalar field they compute is:

$$F(\mathbf{x}) = \sum_{i=1}^N w_i(\mathbf{x}) (d_i(\mathbf{x}) - D_i(\mathbf{x})) \quad (11.16)$$

The zero level of  $F(\mathbf{x})$  is then computed using marching cubes. While this method provides quite accurate results it has a drawback: For a set of depth maps around a closed object, distances from opposite sides interfere with each other. To avoid this effect [12] actually clamps the distance on either side of a depth map. The distance must be left un-clamped far enough behind the depth map so that all distance functions contribute to the zero-level crossing, but not too far so as to compromise the reconstruction of thin structures. This limitation is due to the fact that the method implicitly assumes that the surface has low relief or that there are no self-occlusions. This can be expressed in several ways but perhaps the most intuitive is that every optic ray from every sensor intersects the surface only once. This means that if a point  $\mathbf{x}$  is visible from at least one sensor then it must be visible from all sensors (see Figure 11.12). Using this assumption, an analysis similar to the one in the previous Section leads to some a surprising insight into the algorithm. More precisely, if we set the prior probability for visibility to  $p(V) = 0.5$  and assume the logistic distribution for sensor noise, *i.e.*,

$$p(D_i, D_i^* | \mathcal{I}) \propto \operatorname{sech}\left(\frac{D_i^* - D_i}{2w_i}\right)^2 \quad (11.17)$$

then the probabilistic evidence for  $V$  given all the data exactly corresponds to the right hand side of (11.16). In other words, the sum of signed distance functions of [12] can be seen as an accumulation of probabilistic evidence for visibility of points in space, given a set of noisy measurements of the depth of the 3D scene. This further reinforces the usefulness of probabilistic evidence for visibility.

#### 11.4.6 Deformable Models

In a similar way to the MRF framework in Section 11.4.1, the deformable model framework [27] allows us to search for an optimal surface  $S^*$  that is a minimizer of some user defined energy function  $E$ . In general, this energy will be non-convex with possible local optima. In our case, the optimization problem is posed as follows: find the surface  $S^*$  of  $\mathbb{R}^3$  that minimizes the energy  $E(S)$  defined as:

$$E(S) = E_{ext}(S) + E_{int}(S), \quad (11.18)$$

where  $E_{ext}(S)$  is the external energy term related to the photo-consistency 3D map and  $E_{int}(S)$  is the internal energy term or regularization term, *i.e.*, a smooth prior on the types of surfaces that we expect. Minimizing Eq. (11.18) means finding a surface  $S^*$  such that satisfies the Euler Equation:

$$\nabla E(S^*) = \nabla E_{ext}(S^*) + \nabla E_{int}(S^*) = 0. \quad (11.19)$$

Equation (11.19) establishes the equilibrium condition for an optimal solution and can also be seen as a force balance Equation:

$$\mathbf{F}_{ext}(S^*) + \mathbf{F}_{int}(S^*) = 0 \quad (11.20)$$

with  $\mathbf{F}_{ext}(S) = \nabla E_{ext}(S)$  and  $\mathbf{F}_{int}(S) = \nabla E_{int}(S)$ . A solution to Eq. (11.20) can be found by introducing a time variable  $t$  for the surface  $S$  and solving the following differential Equation:

$$\frac{\partial S}{\partial t} = \mathbf{F}_{ext}(S) + \mathbf{F}_{int}(S). \quad (11.21)$$

The discrete version becomes:

$$S^{k+1} = S^k + \Delta t(\mathbf{F}_{ext}(S^k) + \mathbf{F}_{int}(S^k)). \quad (11.22)$$

Once we have sketched the energies that will drive the process, we need to make a choice for the representation of the surface  $S$ . This representation defines the way the deformation of the surface is performed at each iteration. We choose the triangular mesh because of its simplicity and well known properties, but other options such as implicit surface representations can be used [25].

To completely define the deformation framework, we need an initial value of  $S$ , *i.e.*, an initial surface  $S^0$  that will evolve under the different forces until convergence.  $S^0$  can range from the most basic initial shape such as a bounding box, to a better one like the visual hull, or an even better one such as the provided by the MRF framework in Section 11.4.1.

In the following we describe how to derive the external force from the photo-consistency 3D map and the internal force on a triangular mesh.

#### 11.4.6.1 External Force: Octree-Based Gradient Vector Flow

The external force is directly linked to the photo-consistency 3D map previously described in Section 11.3. We want this force to drive the surface to high photo-consistency locations. However the volume of photo-consistency  $\mathcal{C}(\mathbf{x})$  itself cannot be used as a force to drive the deformable model. A typical force would be the gradient of  $\mathcal{C}(\mathbf{x})$ , *i.e.*,  $\mathbf{F}_{ext}(\mathbf{x}) = \nabla \mathcal{C}(\mathbf{x})$ . The main objection is that it is a very local force defined only in the vicinity of the object surface. A better solution is to use a gradient vector flow (GVF) field derived from the photo-consistency in order drive the deformable model.

The GVF field was introduced by [52] as a way to overcome a difficult problem of traditional deformable models: the capture range of the data term. This problem is caused by the local definition of the force, and the absence of an information propagation mechanism. To eliminate this drawback, and for all the forces derived from the gradient of a scalar field, they proposed to generate a vector field force that propagates the gradient information. The GVF of a scalar field  $f(x, y, z) : \mathbb{R}^3 \mapsto \mathbb{R}$  is defined as the vector field  $\mathbf{F} = [u(x, y, z), v(x, y, z), w(x, y, z)] : \mathbb{R}^3 \mapsto \mathbb{R}^3$  that minimizes the following energy functional  $E_{GVF}$ :

$$E_{GVF} = \int \mu ||\nabla \mathbf{F}||^2 + ||\mathbf{F} - \nabla f||^2 ||\nabla f||^2, \quad (11.23)$$

where  $\mu$  is the weight of the regularization term and  $\nabla \mathbf{F} = [\nabla u, \nabla v, \nabla w]$ . The solution to this minimization problem has to satisfy the Euler Equation:

$$\mu \nabla^2 \mathbf{F} - (\mathbf{F} - \nabla f) ||\nabla f||^2 = 0, \quad (11.24)$$

where  $\nabla^2 \mathbf{F} = [\nabla^2 u, \nabla^2 v, \nabla^2 w]$  and  $\nabla^2$  is the Laplacian operator. A numerical solution can be found by introducing a time variable  $t$  and solving the following differential Equation:

$$\frac{\partial \mathbf{F}}{\partial t} = \mu \nabla^2 \mathbf{F} - (\mathbf{F} - \nabla f) ||\nabla f||^2. \quad (11.25)$$

The GVF can be seen as the original gradient smoothed by the action of a Laplacian operator. This smoothing action allows eliminating strong variations of the gradient and, at the same time, propagating it. The degree of smoothing/propagation is controlled by  $\mu$ . If  $\mu$  is zero, the GVF will be the original gradient, if  $\mu$  is very large, the GVF will be a constant field whose components are the mean of the gradient components. Applied to the deformable model problem, the external force  $\mathbf{F}_{ext}$  is then found as the solution of the following differential Equation:

$$\frac{\partial \mathbf{F}_{ext}}{\partial t} = \mu \nabla^2 \mathbf{F}_{ext} - (\mathbf{F}_{ext} - \nabla \mathcal{C}) ||\nabla \mathcal{C}||^2, \quad (11.26)$$

with  $\mu$  always fixed to 0.1.

#### 11.4.6.2 Mesh Control

The goal of the internal force is to regularize the effect of the external forces. Following the formulation by [10], we define the internal energy  $E_{int}$  of a surface  $S$  as the sum of two terms penalizing for changes in the first and second order derivatives of the surface. A local minimum of the energy  $E_{int}(S)$  satisfies the associated Euler-Lagrange Equation, which gives us the following form for the internal force:

$$\mathbf{F}_{int}(S) == \gamma_1 \Delta S + \gamma_2 \Delta^2 S, \quad (11.27)$$

where  $\Delta$  is the Laplacian operator and  $\Delta^2$  is the biharmonic operator. The discrete version of the Laplacian operator  $\tilde{\Delta}$  on a triangle mesh can be easily implemented using the umbrella operator, *i.e.*, the operator that tries to move a given vertex  $\mathbf{v}$  of the mesh to the center of gravity of its 1-ring neighborhood  $\mathcal{N}_1(\mathbf{v})$ :

$$\tilde{\Delta} \mathbf{v} = \left( \sum_{i \in \mathcal{N}_1(\mathbf{v})} \frac{\mathbf{v}_i}{m} \right) - \mathbf{v}, \quad (11.28)$$

where  $\mathbf{v}_i$  are the neighbors of  $\mathbf{v}$  and  $m$  is the total number of these neighbors (valence). Concerning the discrete version of the biharmonic operator  $\tilde{\Delta}^2$ , its derivation is less trivial:

$$\tilde{\Delta}^2 \mathbf{v} = \frac{1}{1 + \sum_{i \in \mathcal{N}_1(\mathbf{v})} \frac{1}{mm_i}} \tilde{\Delta}(\tilde{\Delta} \mathbf{v}), \quad (11.29)$$

The total internal force on a mesh vertex  $\mathbf{v}$  is defined as:

$$\mathbf{F}_{int}(\mathbf{v}) = \gamma_1 \tilde{\Delta} \mathbf{v} + \gamma_2 \tilde{\Delta}^2 \mathbf{v}. \quad (11.30)$$

Since the texture force  $\mathbf{F}_{ext}$  can sometimes be orthogonal to the surface of the snake, we do not use the force  $\mathbf{F}_{ext}$  itself but its projection  $\mathbf{F}_{ext}^N$  onto the surface normal  $\mathbf{n}$ :

$$\mathbf{F}_{ext}^N(\mathbf{v}) = (\mathbf{n}^\top \cdot \mathbf{F}_{ext}(\mathbf{v})) \mathbf{n}. \quad (11.31)$$

This avoids problems of coherence in the force of neighbor points and helps the internal force to keep a well-shaped surface.

The evolution process (Eq. 11.22) at the  $k^{th}$  iteration can then be written as the evolution of all the points of the mesh  $\mathbf{v}_i$ :

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \Delta t (\mathbf{F}_{ext}^N(\mathbf{v}_i^k) + \mathbf{F}_{int}(\mathbf{v}_i^k)), \quad (11.32)$$

where  $\Delta t$  is the time step and  $\alpha$  is the weight of the regularization term relative to the external term. Equation 11.32 is iterated until convergence of all the points of the mesh is achieved. The time step  $\Delta t$  has to be chosen as a compromise between the stability of the process and the convergence time. An additional step of remeshing is done at the end of each iteration in order to maintain a minimum and a maximum distance between neighbor points of the mesh. This is obtained by a controlled decimation and refinement of the mesh. The decimation is based on the edge collapse operator and the refinement is based on the  $\sqrt{3}$ -subdivision scheme [28].

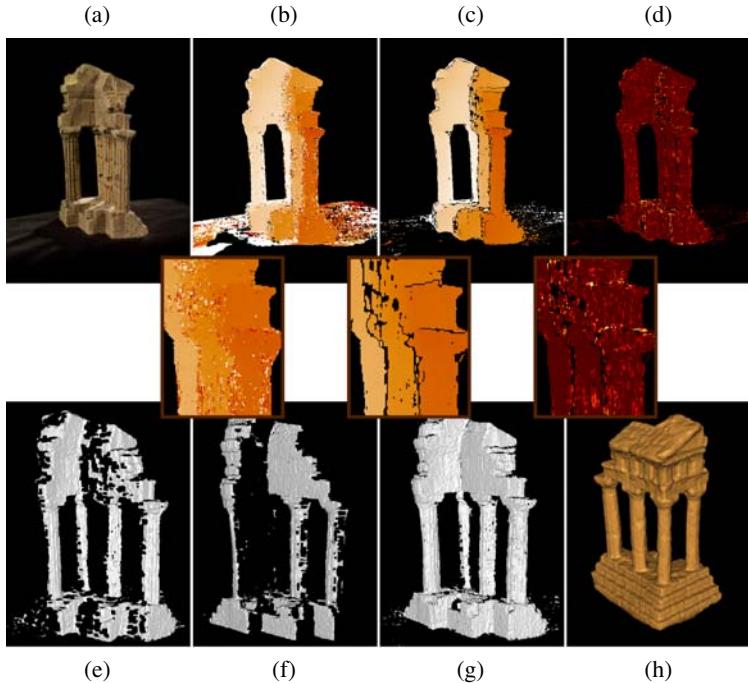
## 11.5 Experiments

### 11.5.1 Depth Map Evaluation

In order to solve the depth-map computation algorithm described in Section 11.3 we use the TRW-S implementation of Kolmogorov [30]. The proposed implementation, running on a 3.0 GHz machine with an nVidia Quadro graphics card, can evaluate 900 NCC depth slices in 20 seconds for the temple sequence (image resolution  $640 \times 480$ ). The TRW-S optimization has a typical run time of 20 seconds for the same images.

For all the experiments we used the following parameter values:  $\beta = 1$ ,  $\lambda = 1$ ,  $\phi_U = 0.04$  and  $\psi_U = 0.002$ . We used an NCC window size of  $5 \times 5$ .

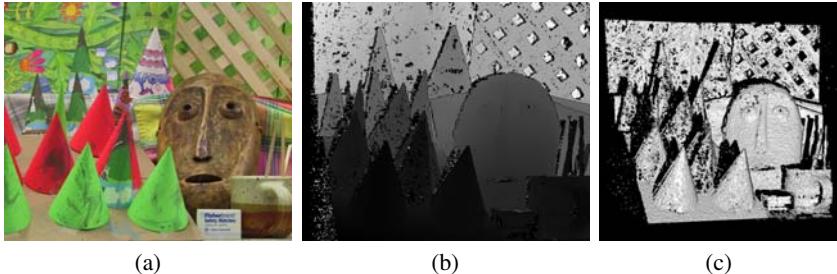
Figure 11.13 illustrates the improvement of the method described in Section 11.3.2 over the voting schemes of [20, 47]. Figure 11.13 (b) shows the depth that would be determined by simply taking the NCC peak with the greatest score. The new method, implemented here with  $K = 9$  peaks, is able to select the peak corresponding to true surface peak from the ranked candidate peaks and Figure 11.13 (d) illustrates that a significant proportion of the true surface peaks are not the absolute



**Fig. 11.13** Results of the depth map estimation algorithm. Two neighboring images are combined with the reference image (a). If we simply took the NCC peak with the maximum score, as in [20], we would obtain (b). The result of the algorithm used in Section 11.3.2 (c) shows a significant reduction in noise. We have corrected noisy estimates of the surface and the *unknown* state has also been used to clearly denote occlusion boundaries and remove poorly matched regions. The number of the correct surface peak returned, ranked by NCC score, is displayed in (d) where dark red indicates the peak with the greatest score. The rendered depth-map is shown in (e) along with the neighboring depth-map (f) with (g) showing the two superimposed. The final reconstruction (h) for the sparse temple sequence (16 images) of [40].

maximum. We also observe that pixels are correctly labeled with the *unknown* state along occlusion boundaries and along areas such as the back wall of the temple and edges of the pillars where the surface normal is oriented away from the camera. Looking at the rendering of this depth-map and its neighbor, Figure 11.13(e-g), we can observe that very few erroneous depths are recovered and we observe that the combination of the two depths maps align and complement each other rather than attempting to fill in the holes on the individual depth-maps which would impact the subsequent multi-view stereo global optimization.

Figure 11.14 shows the results on the ‘cones’ dataset which forms part of the standard dense stereo evaluations images and consists of a single stereo pair with the left image shown. The depth-map again shows a high degree of detail on textured surfaces and we correctly identify occlusion boundaries with the *un-*



**Fig. 11.14** Single view stereo results for the ‘Cones’ data set. The left image of the stereo pair is shown in (a) with the recovered depth-map in (b), rendered in (c)

known state. Further more the algorithm also correctly textures the failure modes of NCC by returning the *unknown* state in texture-less regions where the matching fails to accurately localize the surface.

### 11.5.2 Multi-view Stereo Evaluation

In order to evaluate the improvement of the depth-map estimation algorithm of Section 11.3.2 for multi-view stereo we ran the algorithm on the standard evaluation ‘temple’ dataset. The following table provides the accuracy and completeness measures of [40] against the ground-truth data for the object. In terms of both accuracy and completeness the results provide a significant improvement in both the sparse ring and ring datasets. In particular we observe that the results for the sparse ring offer greater accuracy than the other algorithms [40] running on the ring sequence (3 times as many images) with the exception of [20].

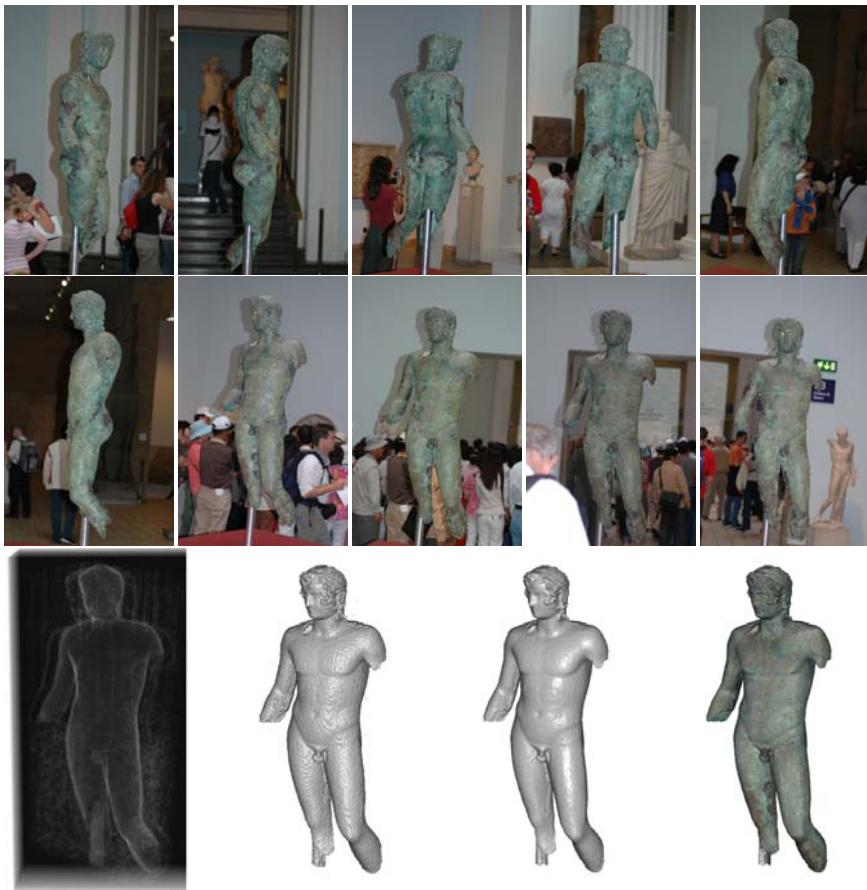
	Accuracy / Completeness		
	Full (312 images)	Ring (47 images)	SparseRing (16 images)
<b>proposed method</b>	<b>0.41mm / 99.9%</b>	<b>0.48mm / 99.4%</b>	<b>0.53mm / 98.6%</b>

### 11.5.3 Digitizing Works of Art

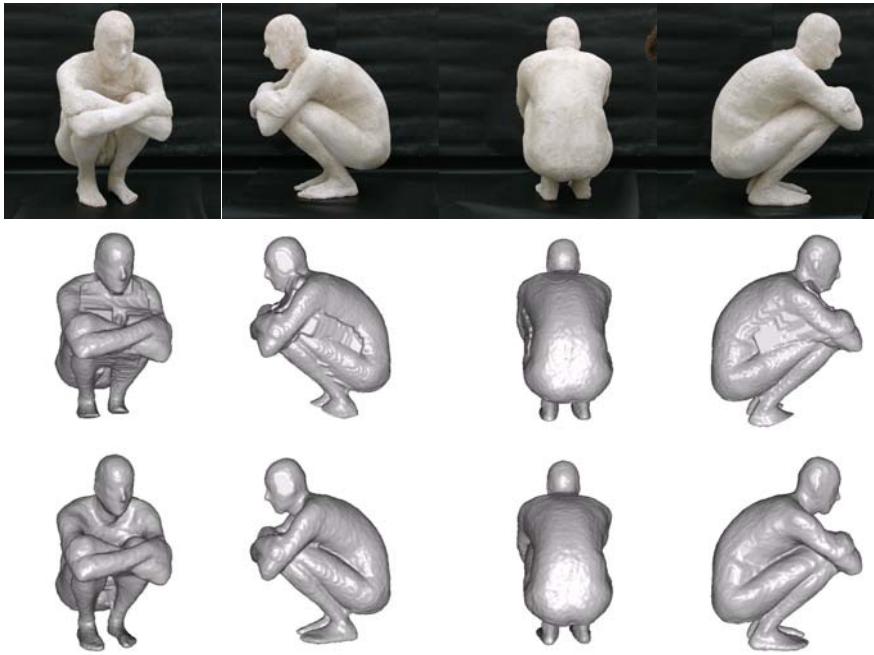
The proposed pipeline has been used to reconstruct a bronze statue located in the British museum in London from holiday photographs. The photographs were taken by a hand held camera during normal visiting hours (see Figure 11.15). This led to the statue being photographed with cluttered and changing background. The camera motion was automatically recovered using a structure-from-motion technique [55]. The bottom row of Figure 11.15 shows the intermediate results obtained while reconstructing the statue. From left to right, we show a rendering of the 3D map of photo-consistency (Section 11.3), the initial surface obtained using graph-cuts (Section 11.4.1), the refined surface obtained with the deformable model (Section 11.4.6), and the same surface textured mapped from the input photographs

using [20]. Note how, even with a noisy photo-consistency 3D volume, the graph-cut solution is able to extract a very detailed surface. However, this surface has discretization artifacts due to the binary nature of the graph-cut solution. These artifacts are completely removed when the surface is refined using a deformable model. A similar refinement step is also used in [17].

We present a second sequence of 72 images of a “*crouching man*” sculpture made of plaster by the modern sculptor Antony Gormley (see Figure 11.16 top).

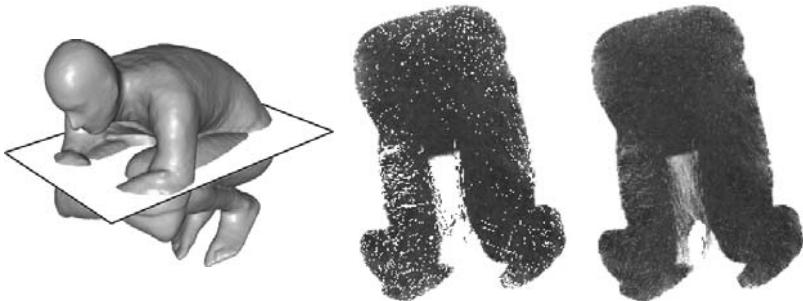


**Fig. 11.15** Statue of a young man, Mimaut Collection. Bronze, Roman copy of the 1st century BC after a Greek original. From Ziphteh, near Tell Atrib (ancient Athribis), Egypt. The sequence was acquired with a hand held camera in the British museum with no special requirements. Background is extremely cluttered. The object of interest is both in the center of the photographs and in focus. Top and middle rows show a few samples of the original sequence. Last row shows from left to right, 3D map of photo-consistency described in Section 11.3, surface extracted using graph-cuts (Section 11.4.1), surface refined using a deformable model (Section 11.4.6) and surface textured-map from the original photographs using [20].



**Fig. 11.16** Comparison of the improvement obtained with the visibility-driven ballooning term. Plaster model of a crouching man by Antony Gormley, 2006. Top: some of the input images. Middle: views of reconstructed model using the technique of [48] with a constant ballooning term. No constant ballooning factor is able to reconstruct correctly the feet and the concavities at the same time. Bottom: views of reconstructed model using the intelligent ballooning proposed by [22] and shown in Figure 11.17 right.

The image resolution is 5 Mpix and the camera motion was recovered by standard structure from motion techniques [55] and further refined using a silhouette-based technique [21]. The object exhibits significant self-occlusions, a large concavity in the chest and two thin legs which make it a very challenging test to validate the new ballooning term. The first step in the reconstruction process is to compute a set of depth-maps from the input images. This process is by far the most expensive of the whole pipeline in terms of computation time. A single depth-map takes between 90 and 120 seconds, the overall computation time being close to 2 hours. Once the depth-maps are computed, a 3D octree grid can be built (see Figure 11.9 left) together with the discontinuity cost and the labeling cost (see Figure 11.9 middle and right respectively). Because of the octree grid, we are able to use up to 10 levels of resolution to compute the graph-cut, *i.e.*, the equivalent of a regular grid of  $1024^3$  voxels. We show in Figure 11.16 some of the images used in the reconstruction (top), the result using an implementation of [48] (middle) and the reconstruction result of the proposed method (bottom). We can appreciate how the constant ballooning term



**Fig. 11.17** Comparison of two different inlier/outlier ratios for the depth sensor noise model. Left: 3D location of one slice of the volume of “evidence of visibility”. Middle: the sensor model is a pure Gaussian without any outlier model. Outliers “drill” tunnels in the visibility volume. Right: the sensor model takes into account an outlier model. The visibility volume is more robust against outliers while the concavities are still distinguishable.

introduced in [48] is unable to reconstruct correctly the feet and the concavities at the same time. In order to recover thin structures such as the feet, the ballooning term needs to be stronger. But even before the feet are fully recovered, the concavities start to over inflate.

Finally we show in Figure 11.17 the effect of having an outlier component in the noise model of the depth sensor when computing the volume of evidence of visibility. The absence of an outlier model that is able to cope with noisy depth estimates appears in the volume of visibility as tunnels “drilled” by the outliers (see Figure 11.17 center). Adding an outlier term clearly reduces the tunneling effect while preserving the concavities (see Figure 11.17 right).

## 11.6 Discussion

We have described a formulation to multi-view stereo that splits the problem into a well defined pipeline of 3 building blocks: camera calibration, computation of a 3D volume of photo-consistency and extraction of a surface from the photo-consistency volume. In this Chapter we have particularly focus on how to compute a 3D volume of photo-consistency, and how to extract a 3D surface from the photo-consistency volume. The main advantages of such an approach are its simplicity and room for improvement, since it uses two very standard off-the-shelf algorithms such as dense stereo and 3D segmentation algorithms. The main disadvantage is the rather simplistic photo-consistency metric, which leads to poor performance in challenging conditions such as sparse set of photographs or poorly textured surfaces. These problems are partially mitigated by explicitly accounting for the failure modes of the window matching technique in Section 11.3. However, a more thorough matching technique using a local planarity assumption such as [17] would also greatly improve results in challenging scenes. The framework we describe in this Chapter has been widely adopted by a variety of multi-view stereo algorithms [7, 8, 18, 20, 24, 29, 38, 42, 47].

This can be mainly justified by the simplicity of the approach, but also by the flexibility that it offers, *e.g.*, when trying to optimally fuse the photo-consistency cue with apparent contours as proposed in [29].

## Appendix: Interpretation of Signed Distance Functions

Using the predicates we have already defined, the assumption of no self-occlusion can be expressed by

$$V \leftrightarrow \forall i V_i. \quad (11.33)$$

From (11.10) and (11.33) we see that if a point  $\mathbf{x}$  is visible (invisible) from one sensor it is visible (invisible) from all sensors, *i.e.*,  $V_1 \leftrightarrow \dots \leftrightarrow V_N \leftrightarrow V$ . Let  $\mathcal{I}$  stand for the prior knowledge which includes the geometric description of the problem and (11.33). Given (11.33) events  $D_1 \dots D_N$  are independent under the knowledge of  $V$  or  $\bar{V}$  which means that using Bayes' theorem we can write:

$$p(V | D_1 \dots D_N \mathcal{I}) = \frac{p(V | \mathcal{I}) \prod_{i=1}^N p(D_i | V \mathcal{I})}{p(D_1 \dots D_N | \mathcal{I})} \quad (11.34)$$

Obtaining the equivalent Equation for  $\bar{V}$  and dividing with Equation (11.34) and taking logs gives us:

$$e(V | D_1 \dots D_N \mathcal{I}) = e(V | \mathcal{I}) + \sum_{i=1}^N \log \frac{p(D_i | V \mathcal{I})}{p(D_i | \bar{V} \mathcal{I})}. \quad (11.35)$$

By several applications of Bayes' theorem we get:

$$e(V | D_1 \dots D_N \mathcal{I}) = \sum_{i=1}^N \log \frac{\alpha_i}{\beta_i} - (N-1)e(V | \mathcal{I}). \quad (11.36)$$

where  $\alpha_i = \int_{d_i}^{\infty} p(D_i, D_i^* | \mathcal{I}) dD_i^*$  and  $\beta_i = \int_0^{d_i} p(D_i, D_i^* | \mathcal{I}) dD_i^*$ . We now set  $e(V | \mathcal{I}) = 0$  and assume the noise model is given by the logistic function

$$p(D_i, D_i^* | \mathcal{I}) \propto \operatorname{sech} \left( \frac{D_i^* - D_i}{2w_i} \right)^2. \quad (11.37)$$

Using standard calculus one can obtain the following expression for the evidence

$$e(V | D_1 \dots D_N \mathcal{I}) = \sum_{i=1}^N w_i (d_i - D_i), \quad (11.38)$$

equal to the average of the distance functions used in [12].

## References

1. Baumgart, B.G.: Geometric modelling for computer vision. Ph.D. thesis, Standford University (1974)
2. Blake, A., Rother, C., Brown, M., Perez, P., Torr, P.: Interactive image segmentation using an adaptive GMMRF model. In: Pajdla, T., Matas, J.(G.) (eds.) ECCV 2004. LNCS, vol. 3021, pp. 428–441. Springer, Heidelberg (2004)
3. Boissonnat, J.D., Faugeras, O., Lebras, E.: Representing stereo data with the delaunay triangulation. *Artificial Intelligence* 44, 41–87 (1990)
4. Bolitho, M., Kazhdan, M., Burns, R., Hoppe, H.: Multilevel streaming for out-of-core surface reconstruction. In: Proceedings of the Eurographics Symposium on Geometry Processing, pp. 69–78 (2007)
5. Boykov, Y., Kolmogorov, V.: Computing geodesics and minimal surfaces via graph cuts. In: Proceedings of the International Conference on Computer Vision, pp. 26–33 (2003)
6. Boykov, Y., Lempitsky, V.: From photohulls to photoflux optimization. In: Proceedings of the British Machine Vision Conference, pp. 1149–1158 (2006)
7. Bradley, D., Boubekeur, T., Heidrich, W.: Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2008)
8. Campbell, N., Vogiatzis, G., Hernández, C., Cipolla, R.: Using multiple hypotheses to improve depth-maps for multi-view stereo. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 766–779. Springer, Heidelberg (2008)
9. Cohen, L.D.: On active contour models and balloons. *CVGIP: Image Understanding* 53(2), 211–218 (1991)
10. Cohen, L.D., Cohen, I.: Finite element methods for active contour models and balloons for 2-D and 3-D images. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 15(11), 1131–1147 (1993)
11. Cornelis, N., Leibe, B., Cornelis, K., Gool, L.: 3d urban scene modeling integrating recognition and reconstruction. *International Journal of Computer Vision* 2-3(78), 121–141 (2008)
12. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proceedings of the ACM SIGGRAPH, pp. 303–312 (1996)
13. Faugeras, O., Keriven, R.: Variational principles, surface evolution, pdes, level set methods and the stereo problem. *IEEE Transactions on Image Processing* 7(3), 335–344 (1998)
14. Favaro, P., Soatto, S.: 3-D Shape Estimation and Image Restoration: Exploiting Defocus and Motion-Blur. Springer, Heidelberg (2007)
15. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Reconstructing building interiors from images. In: Proceedings of the International Conference on Computer Vision (2009)
16. Furukawa, Y., Ponce, J.: Carved visual hulls for image-based modeling. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 564–577. Springer, Heidelberg (2006)
17. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multi-view stereopsis. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
18. Goesele, M., Curless, B., Seitz, S.: Multi-view stereo revisited. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 2402–2409 (2006)
19. Habbecke, M., Kobbelt, L.: A surface-growing approach to multi-view stereo reconstruction. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)

20. Hernández, C., Schmitt, F.: Silhouette and stereo fusion for 3d object modeling. *Computer Vision and Image Understanding* 96(3), 367–392 (2004)
21. Hernández, C., Schmitt, F., Cipolla, R.: Silhouette coherence for camera calibration under circular motion. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 29(2), 343–349 (2007)
22. Hernández, C., Vogiatzis, G., Cipolla, R.: Probabilistic visibility for multi-view stereo. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2007)
23. Hernández, C., Vogiatzis, G., Cipolla, R.: Multi-view photometric stereo. *IEEE Transaction on Pattern Analysis Machine Intelligence* 30(1), 548–554 (2008)
24. Hornung, A., Kobbelt, L.: Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 503–510 (2006)
25. Ilic, S., Fua, P.: Implicit meshes for surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(2), 328–333 (2006)
26. Jaynes, E.: *Probability Theory, The Logic of Science*. Cambridge University Press, Cambridge (2003)
27. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. *International Journal of Computer Vision* 1, 321–332 (1988)
28. Kobbelt, L.:  $\sqrt{3}$ -subdivision. In: *Proceedings of the ACM SIGGRAPH*, pp. 103–112 (2000)
29. Kolev, K., Cremers, D.: Integration of multiview stereo and silhouettes via convex functionals on convex domains. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) *ECCV 2008, Part I*. LNCS, vol. 5302, pp. 752–765. Springer, Heidelberg (2008)
30. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 28(10), 1568–1583 (2006)
31. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 26(2), 147–159 (2004)
32. Kutulakos, K.N., Seitz, S.M.: A theory of shape by space carving. *International Journal of Computer Vision* 38(3), 199–218 (2000)
33. Lempitsky, V., Boykov, Y., Ivanov, D.: Oriented visibility for multiview reconstruction. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) *ECCV 2006*. LNCS, vol. 3953, pp. 226–238. Springer, Heidelberg (2006)
34. Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J., Fulk, D.: The digital michelangelo project: 3d scanning of large statues. In: *Proceedings of the ACM SIGGRAPH*, pp. 15–22 (2000)
35. Lhuillier, M., Quan, L.: A quasi-dense approach to surface reconstruction from uncalibrated images. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 27(3), 418–433 (2005)
36. Goesele, M., Snavely, N., Curless, B., Hoppe, H., Seitz, S. M.: Multi-view stereo for community photo collections. In: *Proceedings of the International Conference on Computer Vision* (2007)
37. Pollefeys, M., Gool, L.J.V., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual modeling with a hand-held camera. *International Journal of Computer Vision* 59(3), 207–232 (2004)

38. Pollefeys, M., Nistér, D., Frahm, J.M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewénius, H., Yang, R., Welch, G., Towles, H.: Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision* 78(2-3), 143–167 (2008)
39. Pons, J.P., Keriven, R., Faugeras, O.: Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *The International Journal of Computer Vision* 72(2), 179–193 (2007)
40. Seitz, S., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 519–528 (2006)
41. Sinha, S., Pollefeys, M.: Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In: *Proceedings of the International Conference on Computer Vision*, pp. 349–356 (2005)
42. Sinha, S.N., Mordohai, P., Pollefeys, M.: Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In: *Proceedings of the International Conference on Computer Vision* (2007)
43. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: Exploring image collections in 3d. In: *Proceedings of the ACM SIGGRAPH* (2006)
44. Steger, E., Kutulakos, K.N.: A theory of refractive and specular 3d shape by light-path triangulation. *International Journal of Computer Vision* 76(1) (2008)
45. Strecha, C., von Hansen, W., Gool, L.V., Fua, P., Thoennesen, U.: On benchmarking camera calibration and multi-view stereo for high resolution imagery. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2008)
46. Tran, S., Davis, L.: 3d surface reconstruction using graph cuts with surface constraints. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) *ECCV 2006. LNCS*, vol. 3952, pp. 218–231. Springer, Heidelberg (2006)
47. Vogiatzis, G., Hernández, C., Torr, P., Cipolla, R.: Multi-view stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 29(12), 2241–2246 (2007)
48. Vogiatzis, G., Torr, P., Cipolla, R.: Multi-view stereo via volumetric graph-cuts. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 391–398 (2005)
49. Weise, T., Leibe, B., Gool, L.V.: Fast 3d scanning with automatic motion compensation. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2007)
50. Witkin, A.: Recovering surface shape and orientation from texture. *Artificial Intelligence* 17(1-3), 17–45 (1981)
51. Woodham, R.: Photometric method for determining surface orientation from multiple images. *Optical Engineering* 19(1), 139–144 (1980)
52. Xu, C., Prince, J.L.: Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 359–369 (1998)
53. Zebedin, L., Bauer, J., Karner, K., Bischof, H.: Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 873–886 (2008)
54. Zhang, L., Snavely, N., Curless, B., Seitz, S.M.: Spacetime faces: High-resolution capture for modeling and animation. In: *ACM Annual Conference on Computer Graphics*, pp. 548–558 (2004)
55. Zisserman, A., Hartley, R.: *Multiple View Geometry*. Springer, Heidelberg (2000)

# Chapter 12

## Practical 3D Reconstruction Based on Photometric Stereo

George Vogiatzis and Carlos Hernández

**Abstract.** Photometric Stereo is a powerful image based 3D reconstruction technique that has recently been used to obtain very high quality reconstructions. However, in its classic form, Photometric Stereo suffers from two main limitations: Firstly, one needs to obtain images of the 3D scene under multiple different illuminations. As a result the 3D scene needs to remain static during illumination changes, which prohibits the reconstruction of deforming objects. Secondly, the images obtained must be from a single viewpoint. This leads to depth-map based 2.5 reconstructions, instead of full 3D surfaces. The aim of this Chapter is to show how these limitations can be alleviated, leading to the derivation of two practical 3D acquisition systems: The first one, based on the powerful Coloured Light Photometric Stereo method can be used to reconstruct moving objects such as cloth or human faces. The second, permits the complete 3D reconstruction of challenging objects such as porcelain vases. In addition to algorithmic details, the Chapter pays attention to practical issues such as setup calibration, detection and correction of self and cast shadows. We provide several evaluation experiments as well as reconstruction results.

### 12.1 Introduction

Photometric stereo is a well established 3D reconstruction technique based on the powerful shading cue. A sequence of images (typically three or more) of a 3D scene are obtained from the same viewpoint and under varying illumination. From the intensity variation in each pixel one can estimate the local orientation of the surface

---

George Vogiatzis  
Aston University, Birmingham, UK  
e-mail: [g.vogiatzis@aston.ac.uk](mailto:g.vogiatzis@aston.ac.uk)

Carlos Hernández  
Toshiba Research Cambridge, UK  
e-mail: [carlos.hernandez@crl.toshiba.co.uk](mailto:carlos.hernandez@crl.toshiba.co.uk)

that projects onto that pixel. By integrating all these surface orientations, a very detailed estimate of the surface geometry can be obtained. Photometric stereo can also provide the surface reflectance properties as part of the same process.

The first reference for photometric stereo is [5]. In early papers (see [22]) the surface reflectance model was constrained to be Lambertian, an assumption that considerably simplifies calculations. Photometric stereo was subsequently relaxed to non-Lambertian reflectance models (e.g., [30] [32] [34]) but the full potential of the technique was only recently demonstrated with works such as [14] [29] [44] that obtained reconstructions of spectacular accuracy. Furthermore, in recent work [33] photometric stereo was shown to be able to significantly refine reconstruction results obtained by 3D laser range scanners. However, the method in its classic formulation suffers from some key limitations

- To obtain a reconstruction one must photograph the object in the same pose several times under changing illumination. This makes it very difficult to reconstruct a moving or deforming object during its motion.
- All photographs must be taken from a single view-point. This restricts reconstructions to 2.5D depth-maps and precludes the full reconstruction in-the-round of a closed 3D surface.

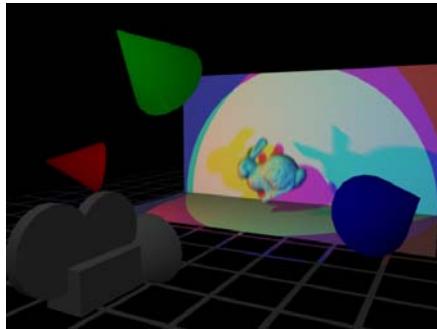
In this Chapter we describe two advances in the state-of-the-art of Photometric stereo that alleviate these two limitations. Firstly we show how a coloured-light photometric stereo variant can be used to obtain independent reconstructions of the object with each photograph obtained. This makes it trivial to use the technique to reconstruct deforming objects such as moving cloth or faces. Second, we describe an elegant generalisation of photometric-stereo to multiple view-points. This method can obtain very accurate closed-surface reconstructions of objects in-the-round such as sculpture.

## 12.2 Photometric Stereo with Coloured Light

To motivate this work, consider the problem of obtaining a dynamic 3D model of a deforming object such as cloth or a human face. This topic has received considerable attention in recent literature [38] [39] [41] [48] [49]. The complexity underlying the simplest of cloth and facial motions motivates capturing geometry and motion data from the real world.

Existing algorithms one might employ for capturing detailed 3D models of deforming objects include multiple view stereo [42], photometric stereo [20] [46], and laser based methods [28]. However, most of these techniques require that the subject stand still during the acquisition process, or move slowly [31].

This Section describes a practical technique for acquiring complex motion data from real objects such as cloth or a face. The required setup consists of an ordinary video camera and three coloured light sources (see Figure 12.1). The key observation is that in an environment where red, green, and blue light is emitted from different directions, a Lambertian surface will reflect each of those colours



**Fig. 12.1** Setup. A schematic representation of our multispectral setup.

simultaneously without any mixing of the frequencies [37]. The quantities of red, green and blue light reflected are a linear function of the surface normal direction. A colour camera can measure these quantities, from which an estimate of the surface normal direction can be obtained. By applying this technique to a video sequence of a deforming object, one can obtain a sequence of normal maps for that object which are integrated to produce a sequence of depth-maps. In essence this technique can be seen as a variant of classic three-source photometric stereo. We will now give a brief overview of that technique and then explain how it is related to coloured photometric stereo. We will then explain in more detail some of the practical aspects of the method including calibration, how it compares numerically to ordinary photometric stereo, and how to cope with shadows.

### 12.2.1 Classic Three-Source Photometric Stereo

In classic three-source photometric stereo we are given three images of a scene, taken from the same viewpoint, and illuminated by three distant light sources. The light sources emit the same light frequency spectrum from three different non-coplanar directions. We will assume an orthographic camera (with infinite focal length) for simplicity, even though the extension to the more realistic projective case is straightforward [43]. In the case of orthographic projection one can align the world coordinate system so that the  $xy$  plane coincides with the image plane while the  $z$  axis corresponds to the viewing direction. The surface in front of the camera can then be parametrized as a height function  $z(x,y)$ . If  $\nabla z$  is the gradient of the function with respect to  $x$  and  $y$ , one can define the vector

$$\mathbf{n} = \frac{1}{\sqrt{1 + |\nabla z|^2}} \begin{pmatrix} \nabla z \\ -1 \end{pmatrix}$$

that is locally normal to the surface at  $(x,y)$ . We can also define a 2D projection operator  $\mathcal{P}[\mathbf{x}] = (x_1/x_3, x_2/x_3)$  so that it follows that  $\nabla z = \mathcal{P}[\mathbf{n}]$ .

Now for  $i = 1, \dots, 3$  let  $c_i(x, y)$  denote the pixel intensity of pixel  $(x, y)$  in the  $i$ -th image. We assume that, in the  $i$ -th image, the surface point  $(x, y, z(x, y))^\top$  is illuminated by a distant light source whose direction is denoted by the vector  $\mathbf{l}_i$  and whose spectral distribution is  $E_i(\lambda)$ . We also assume that the surface point absorbs incoming light of various wavelengths according to the reflectance function  $R(x, y, \lambda)$ . Finally, let the response of the camera sensor at each wavelength be given by  $S(\lambda)$ . Then the pixel intensity  $c_i(x, y)$  is given by [37]

$$c_i(x, y) = (\mathbf{l}_i^\top \mathbf{n}) \int E(\lambda) R(x, y, \lambda) S(\lambda) d\lambda. \quad (12.1)$$

The value of this integral is known as the surface *albedo*  $\rho$  so that (12.1) becomes a simple dot product

$$c_i = \mathbf{l}_i^\top \rho \mathbf{n}. \quad (12.2)$$

Photometric stereo methods use the linear constraints of (12.2) to solve for  $\rho \mathbf{n}$  in a least squares sense. From this they obtain the gradient of the height function  $\nabla z = \mathcal{P}[\rho \mathbf{n}]$  which is then integrated to produce the function  $z$  itself. In three-source photometric stereo, when the point is not in shadow with respect to all three lights, we measure three positive intensities  $c_i$ , each of which gives a constraint on  $\rho \mathbf{n}$ . If we write  $L = [\mathbf{l}_1 \mathbf{l}_2 \mathbf{l}_3]^\top$  and  $\mathbf{c} = [c_1 c_2 c_3]^\top$  then the system has exactly one solution which is given by

$$\rho \mathbf{n} = L^{-1} \mathbf{c}. \quad (12.3)$$

### 12.2.2 Multi-spectral Sources and Sensors

This Section provides the link between classic three source photometric stereo and the multi-spectral/multi-sensor case. We follow the exposition of [25]. In colour photometric stereo each of the three camera sensors (Red, Green and Blue) can be seen as a linear combination of the three images of a classic photometric stereo acquisition. To see this, consider the pixel intensity of pixel  $(x, y)$  for the  $i$ -th sensor, given by

$$c_i(x, y) = \sum_j (\mathbf{l}_j^\top \mathbf{n}) \int E_j(\lambda) R(x, y, \lambda) S_i(\lambda) d\lambda. \quad (12.4)$$

Note that as opposed to Eq. (12.1) the sensor sensitivity  $S_i$  and spectral distribution  $E_j$  are different per sensor and per light source respectively. To be able to determine a unique mapping between RGB values and normal orientation we need to assume a monochromatic surface. We therefore require that  $R(x, y, \lambda) = \rho(x, y) \alpha(\lambda)$ . Where  $\rho(x, y)$  is the monochromatic albedo of the surface point and  $\alpha(\lambda)$  is the characteristic chromaticity of the material. Let

$$v_{ij} = \int E_j(\lambda) \alpha(\lambda) S_i(\lambda) d\lambda$$

and

$$\mathbf{v}_j = (v_{1j} \ v_{2j} \ v_{3j})^\top.$$

Then the vector of the three sensor responses at a pixel is given by

$$\mathbf{c} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \mathbf{v}_3] [\mathbf{l}_1 \ \mathbf{l}_2 \ \mathbf{l}_3]^\top \rho \mathbf{n}.$$

Essentially each vector  $\mathbf{v}_j$  provides the response measured by the three sensors when a unit of light from source  $j$  is received by the camera. If matrix  $[\mathbf{v}_1 \ \mathbf{v}_2 \ \mathbf{v}_3]$  is known, then we can compute

$$\hat{\mathbf{c}} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \mathbf{v}_3]^{-1} \mathbf{c}.$$

The values of  $\hat{\mathbf{c}}$  can be treated in exactly the same way as the three gray-scale images of Section 12.2.1. The next Section describes a simple process for calibrating colour photometric stereo.

### 12.2.3 Calibration

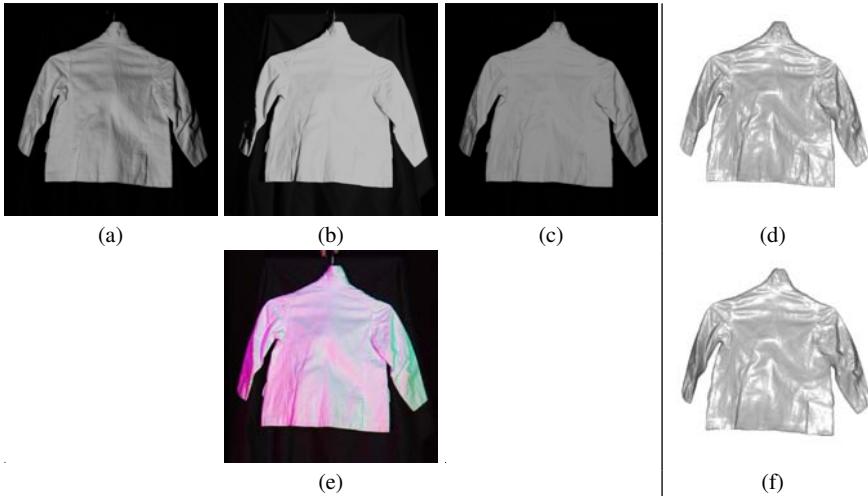
In [19] the authors propose a simple scheme for calibrating objects that can be flattened and placed on a planar board. The system detects special patterns on the board, from which it can estimate its orientation relative to the camera. By measuring the RGB response corresponding to each orientation of the material they estimate the entire matrix

$$M = [\mathbf{v}_1 \ \mathbf{v}_2 \ \mathbf{v}_3] [\mathbf{l}_1 \ \mathbf{l}_2 \ \mathbf{l}_3]^\top$$

that links the normals to RGB triplets. Here we propose a two-step process. Firstly, we use a mirror sphere to estimate light directions  $\mathbf{l}_1$ ,  $\mathbf{l}_2$  and  $\mathbf{l}_3$ . This is a standard process which has been applied in a number of photometric stereo methods. Secondly, we capture three sequences of the object moving in front of the camera. In each sequence, we switch on only one of the three lights at a time. In the absence of noise and if the monochromatic assumption was satisfied, the RGB triplets we acquired would be multiples of  $\mathbf{v}_j$  when light  $j$  was switched on. We therefore do a least squares fit to the three sets of RGB to get the directions of  $\mathbf{v}_1$ ,  $\mathbf{v}_2$  and  $\mathbf{v}_3$ . To get the relative lengths of the three vectors we can use the ratios of the lengths of the RGB vectors. The length of  $\mathbf{v}_j$  is set to the maximum length in RGB space, of all the triplets when light  $j$  was switched on.

### 12.2.4 Comparison with Photometric Stereo

To evaluate the accuracy of the per-frame depth-map estimation we reconstructed a static object (a jacket) using classic photometric stereo with three images each taken under different illumination. The same object was reconstructed using a single image, captured under simultaneous illumination by three coloured lights, using our technique. Figure 12.2 shows the two reconstructions side by side. The results look very similar and the average distance between the two meshes is only 1.4% of



**Fig. 12.2** Comparison with photometric stereo. (a-c) show three grayscale images captured by a digital camera, each taken under a different illumination, providing the input to a classic photometric stereo reconstruction [51] shown in (d). (e) shows a frame from a jacket sequence, where the same object is illuminated *simultaneously* by three different coloured lights. Our algorithm only uses one such frame to generate the surface mesh shown in (f). Note that both algorithms give very similar results, but only the new one (bottom row) can be applied to video since only one frame is required to obtain a reconstruction. As a quantitative comparison, the average error between both reconstructions is only **1.4%** of the bounding box diagonal.

the bounding box diagonal. It is worth noting that even though photometric stereo achieves comparable accuracy, it cannot be used on a non-static object whose shape will change while the three different images are captured. Since our method only uses one image, it is suitable for obtaining frame-by-frame reconstructions of a de-forming object.

### 12.2.5 The Problem of Shadows

One of the most important challenges for all photometric reconstruction methods is the frequent presence of shadows in an image. No matter how careful the arrangement of the light sources, cast or self shadows are an almost unavoidable phenomenon, especially in objects with complex geometries. This Section examines in detail the phenomenon of shadows in photometric stereo with three light sources.

Shadows in photometric stereo have been the topic of a number of papers [2, 6, 11]. Most papers assume we are given four or more images under four different illuminations. This over-determines the local surface orientation and albedo (3 degrees of freedom) which implies that we can use the residual of some least squares solution, to determine whether shadowing has occurred. However when we

are only given three images, as in the case of colour photometric stereo there are no *sparse* constraints against which to test our hypothesis. Therefore the problem of detecting shadows becomes more difficult. Furthermore, when a pixel is in shadow in one of the three images most methods simply discard it. Here we show how one can use the remaining two image intensity measurements to estimate the surface geometry inside the shadow region. Using an argument based purely on counting degrees of freedom and Equations, this is theoretically possible since we need to estimate 2 DOF per pixel (depth and albedo) and we have two independent measurements per pixel. The solution is based on enforcing (1) integrability of the gradient field, as well as (2) smoothness in the recovered shape.

Consider a three-source photometric stereo setup where one point is in shadow, say in the 3-rd image. This implies that the image measurement of  $c_3$  cannot be used as a constraint. Since each equation (12.2) describes a 3D plane, the intersection of the two remaining constraints is a 3D line given by

$$(c_2\mathbf{l}_1 - c_1\mathbf{l}_2)^\top \mathbf{n} = 0, \quad (12.5)$$

or equivalently

$$\mathcal{P}[(c_2\mathbf{l}_1 - c_1\mathbf{l}_2)]^\top \nabla z = 1. \quad (12.6)$$

This Equation was derived by [50] and used for stereo matching in a two-view photometric stereo setup, and subsequently used by [12] to perform uncalibrated photometric stereo and by [7] in their proof of non-existence of a general illumination invariant. Here we show how this Equation can be used in a least squares framework to perform three-source photometric stereo in the presence of shadows.

### 12.2.5.1 Integrating in the Shadowed Regions

According to the image constraints and assuming no noise in the data, we can have one of the following three cases:

1. The surface point is in shadow in two or more images. In this case there is no constraint in  $\nabla z$  from the images.
2. The surface point is not in shadow in any of the three images. In this case  $\nabla z$  coincides with  $\mathcal{P}[L^{-1}\mathbf{c}]$ .
3. The surface point is in shadow in exactly one image, say the 3rd. In this case  $\nabla z$  must lie on the line  $\mathcal{P}[(c_2\mathbf{l}_1 - c_1\mathbf{l}_2)]^\top \nabla z = 1$ . We call this line the *shadow line* of the shaded pixel.

Now in the presence of noise in the data  $\mathbf{c}$ , cases 2 and 3 above do not hold exactly as  $\mathcal{P}[L^{-1}\mathbf{c}]$  and  $\mathcal{P}[(c_2\mathbf{l}_1 - c_1\mathbf{l}_2)]$  are corrupted. The estimation of the unknown height function  $z$  becomes a least squares problem with two different data terms, one for pixels under shadow and another one for pixels seen in all three images.

Under noise in the image data  $\mathbf{c}$ , the 2D point  $\mathcal{P}[L^{-1}\mathbf{c}]$  and 2D line  $\mathcal{P}[(c_2\mathbf{l}_1 - c_1\mathbf{l}_2)]$  are not perfectly consistent with the model. For non-shadowed pixels, the difference between model and data can be measured by the *point-to-point* square difference term

$$\mathcal{E} = |\nabla z - \mathcal{P}[L^{-1}\mathbf{c}]|^2. \quad (12.7)$$

In the case of the shadowed pixels we have a *point-to-line* square difference term

$$\bar{\mathcal{E}}^{(3)} = (\mathcal{P}[(c_2\mathbf{l}_1 - c_1\mathbf{l}_2)]^\top \nabla z - 1)^2. \quad (12.8)$$

Assume we are given a labelling of pixels into all the possible types of shadow. Let  $\mathcal{S}$  contain all non-shadowed pixels while  $\mathcal{S}_i$  contains pixels shaded in the  $i$ -th image. Our cost function becomes

$$\sum_{j \in \mathcal{S}} \mathcal{E}_j + \sum_{j \in \mathcal{S}_1} \bar{\mathcal{E}}_j^{(1)} + \sum_{j \in \mathcal{S}_2} \bar{\mathcal{E}}_j^{(2)} + \sum_{j \in \mathcal{S}_3} \bar{\mathcal{E}}_j^{(3)}$$

which is a set of quadratic terms in  $\nabla z$  and thus  $z$ . Finding the minimum of this quantity is a simple unconstrained linear least squares problem that can be solved using a sparse linear solver such as UMFPACK [9].

Figure [12.3] shows this idea applied in practise on synthetic data. It provides evidence that in its present form the problem is ill-conditioned, especially in larger shadowed regions (see Figure [12.3c]). The following Section sheds more light on this and describes our proposed remedy (see figures [12.3d] and [12.3e]).

### 12.2.5.2 Regularisation in the Shadow Regions

The linear least squares optimisation framework described in Section [12.2.1] when executed in practise shows signs of ill-posedness in the presence of noise. This is demonstrated in the synthetic case of figure [12.3] where three images of a sphere have been generated. Three shadow regions corresponding to each of the three lights have been introduced. Even though the overall shape of the object is accurately captured, some characteristic ‘scratch’ artifacts are observed. These are caused by the point-to-line distances which do not introduce enough constraints in the cost function. The point  $\nabla z$  can move significantly in a direction parallel to the corresponding shadow line only to gain a slight decrease in the overall cost. This results in violent perturbations in the resulting height function that manifest themselves as deep scratches that follow the 2D flow  $\mathcal{P}[(c_2\mathbf{l}_1 - c_1\mathbf{l}_2)]$ .

If we push the analysis even further and have one of the images completely shadowed, we then fall back to the two-source photometric stereo setup shown in Figure [12.4]. When only two images are available without shadow (see Figure [12.4] top), after factoring out the albedo [12.5] we can only determine the depth gradient along specific directions for each pixel  $\mathcal{P}[(c_2\mathbf{l}_1 - c_1\mathbf{l}_2)]$ . If we look at these directions as a vector field, then depth can be computed independently along each streamline or “characteristic curve” (see Figure [12.4b]). In other words, there is no constraint between the depth of two characteristic curves and one pixel can only belong to a single characteristic curve. After integrating every characteristic curve independently (see Figure [12.4c]), we obtain a possible reconstruction that is different from the original true shape, but that perfectly agrees with the given constraints. In order

to choose one among the possible solutions, some type of regularisation is needed (see Figure 12.4d and 12.4e).

Regularisation can be seen as a prior on the type of solutions we expect. In order to better understand what types of prior might be relevant, let us restate the problem assumptions. We have a three source photometric stereo setup with varying albedo, and one of the lights is occluded, *i.e.*, we locally have a two source photometric stereo setup with varying albedo. From the theory we know that in the photometric stereo setup, the albedo and the geometry are coupled, and if there is not enough data available, both are indistinguishable. This coupling exactly indicates what two types of priors one might use: either a shape smoothness prior favouring smooth shapes or an albedo smoothness prior favouring smooth albedo. The exact type of prior used should depend on the type of data captured. A good regularising criterion must satisfy two main requirements:

- The scheme must be consistent with the linear least squares framework. No non-linear constraints can be enforced.
- It must suppress noise while preserving as much of the data as possible.

In the following we describe two different regularisation schemes that favour smooth shapes while preserving the data as much as possible. Their main difference is that one favours shapes with a *smooth shading* under the occluded light, while the second one favours *smooth shapes*. The second scheme can be used in a two-source photometric stereo setup as it is independent of the occluded third light (see Figure 12.4).

### Shading Regularisation

In this approach we want to impose regularisation on the collected shading intensities, thereby "inpainting" [4] the shadowed regions in order to recover the intensities we would collect had the light not been occluded and the albedo been constant. From Equation (12.3) we can parametrize the shadow line as a function of the missing shading  $\mu$

$$\nabla z = \mathcal{P} \left[ L^{-1} \begin{pmatrix} c_1 \\ c_2 \\ 0 \end{pmatrix} + \mu \rho L^{-1} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right] \quad (12.9)$$

This parameter represents the value  $\mathbf{I}_3^\top \mathbf{n}$  would have, had the point not been in shadow in the 3-rd image. In order to simplify the notation of (12.9) we define matrix  $M = L^{-1}$  where vector  $\mathbf{m}_i$  is the  $i^{\text{th}}$  column of matrix  $M$ , giving

$$\nabla z = \mathcal{P}[c_1 \mathbf{m}_1 + c_2 \mathbf{m}_2 + \mu \rho \mathbf{m}_3] \quad (12.10)$$

We observe that, because  $c_1$  and  $c_2$  already encode the albedo  $\rho$  in, Equation (12.10) is in fact independent of  $\rho$  due to the projection operator. We also note that  $\nabla z$  is not a linear function of  $\mu$  meaning that we cannot directly regularise the missing shading  $\mu$  in a linear least squares framework. However, we can perform a change of variables and introduce a new variable  $w$  per shaded pixel

$$w(\mu) = \frac{\mathbf{e}_3^\top (c_1 \mathbf{m}_1 + c_2 \mathbf{m}_2)}{\mathbf{e}_3^\top (c_1 \mathbf{m}_1 + c_2 \mathbf{m}_2) + \mu \rho \mathbf{e}_3^\top \mathbf{m}_3}, \quad (12.11)$$

with  $\mathbf{e}_3 = (0, 0, 1)^\top$ . The new variable  $w$  still specifies a location along the shadow line of that pixel so Equation (12.10) simply becomes

$$\nabla z = w \mathcal{P}[c_1 \mathbf{m}_1 + c_2 \mathbf{m}_2] + (1 - w) \mathcal{P}[\mathbf{m}_3] \quad (12.12)$$

The term is now quadratic with respect to  $\nabla z$  and  $w$ , allowing us to regularise the solution in a meaningful way by using first order  $|\nabla w|$  and second order  $|\nabla^2 w|$  regularisation terms on  $w$ . The point-to-line distance of (12.8) can now be replaced with the following point-to-point distance

$$\bar{\mathcal{E}}^{(3)} = |\nabla z - w \mathcal{P}[c_1 \mathbf{m}_1 + c_2 \mathbf{m}_2] - (1 - w) \mathcal{P}[\mathbf{m}_3]|^2 + \alpha |\nabla w|^2 + \beta |\nabla^2 w|^2, \quad (12.13)$$

where  $\alpha$  and  $\beta$  are regularisation weights. As  $w$  is a proxy for  $\mu$ , this corresponds to introducing smoothness in the product  $\mathbf{l}_3^\top \mathbf{n}$ . We can therefore eliminate the scratch artifacts while letting  $\mathbf{n}$  have variability in the directions perpendicular to  $\mathbf{l}_3$ .

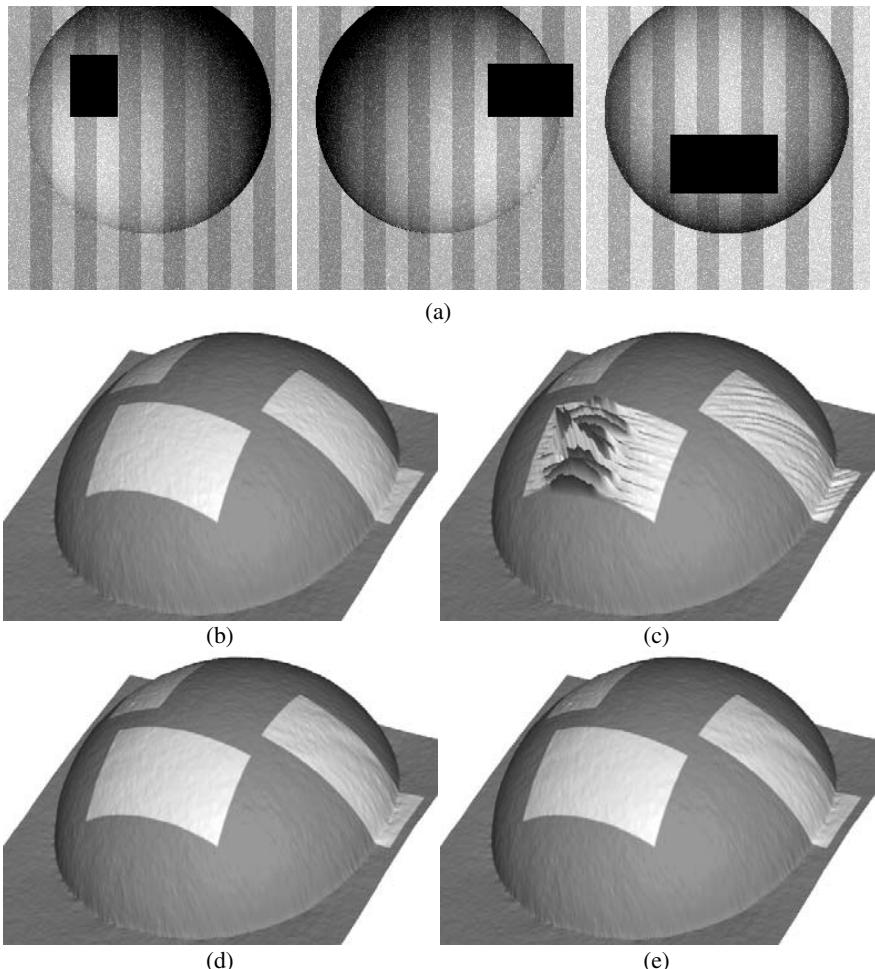
## Shape Regularisation

The most common way of regularising shape is using first-order and second-order regularisation terms. In the context of a height field, this is achieved by minimising the norm of the gradient of the height field  $|\nabla z|$  or minimising the Laplacian of the height field  $|\nabla^2 z|$ . The latter is known to have good noise reduction properties and to produce smooth well behaved surfaces with low curvature. However, both the gradient and the Laplacian are isotropic so they tend to indiscriminately smooth along all possible directions. See [1] for a good discussion of anisotropic alternatives to Laplacian filtering in the context of gradient field integration. In the context of our problem, there is an efficient way of achieving anisotropic versions of both the first-order and the second-order regularisation terms. From Equation (12.6), we observe that the shape is totally unconstrained along perpendicular directions to  $\mathcal{P}[(c_2 \mathbf{l}_1 - c_1 \mathbf{l}_2)]$ . The directions  $\mathcal{P}[(c_2 \mathbf{l}_1 - c_1 \mathbf{l}_2)]$  define characteristic curves, visually showing the constraint induced by the two lights (see Figure 12.4b). Therefore a good way of regularising the shape is along perpendicular directions  $\mathbf{u}$  to the characteristic curves, *i.e.*,  $\mathbf{u}^\top \mathcal{P}[(c_2 \mathbf{l}_1 - c_1 \mathbf{l}_2)] = 0$ . The point-to-line distance term (12.8) is therefore extended with anisotropic first and second order regularisation terms

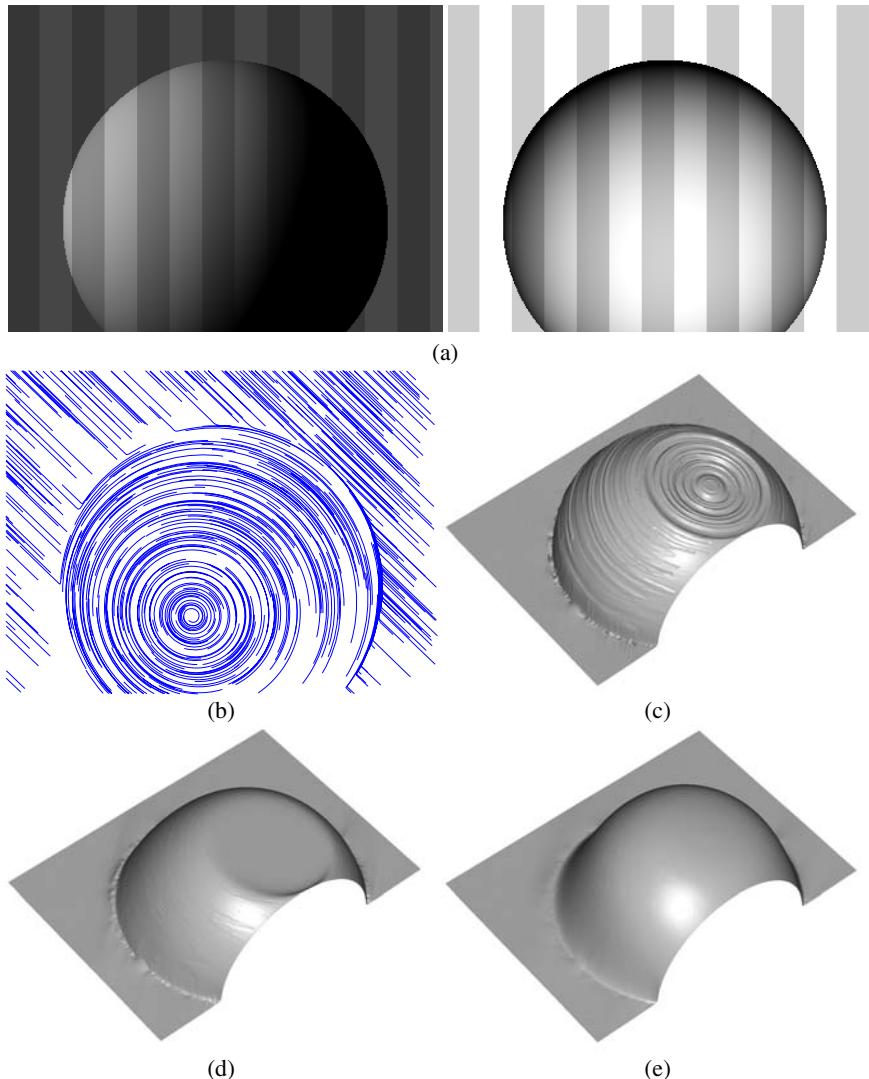
$$\bar{\mathcal{E}}^{(3)} = (\mathcal{P}[(c_2 \mathbf{l}_1 - c_1 \mathbf{l}_2)]^\top \nabla z - 1)^2 + \alpha |\mathbf{u}^\top \nabla z|^2 + \beta |\mathbf{u}^\top H(z) \mathbf{u}|^2, \quad (12.14)$$

$\alpha$  and  $\beta$  being the regularisation weights and  $H(z)$  the Hessian matrix.

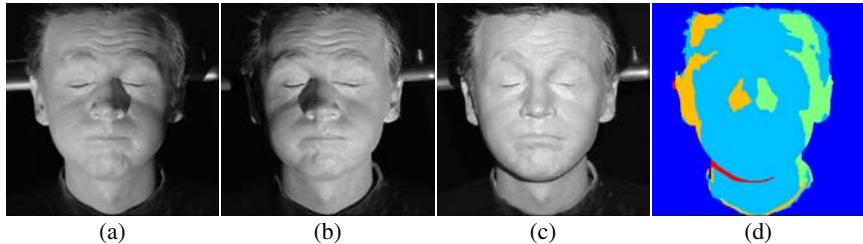
Throughout all of the previous discussion we have assumed knowledge of labelling of pixels according to shadows. The next Section discusses how we propose to segment shadow regions in the image.



**Fig. 12.3** Regularization schemes. This is an experiment on a synthetic sphere designed to validate the proposed regularisation constraints. (a) shows the input images where the black rectangles correspond to occluded regions. This object is illuminated from three directions and the three white regions are occluded in the corresponding images. Middle row shows the photometric stereo solution without shadows (b) and the effect of optimising the surface with no regularisation at all, *i.e.*, just using integrability (c). Note the characteristic ‘scratch’ artifacts. (d) shows the resulting surface after adding a shading regularisation term with optimal values  $\alpha = 6.1, \beta = 0$ . (e) shows the resulting surface after adding a shape regularisation term with optimal values  $\alpha = 0.08, \beta = 0.46$ . See Section 12.2.5.2 for a description of the algorithms. The artifacts have been suppressed while the data has been preserved unsmoothed. Note how both regularisation schemes give almost identical results.



**Fig. 12.4** Two-source varying albedo photometric stereo setup. In this experiment we show a two-source photometric stereo with varying albedo. (a) shows the two input images. (b) shows the characteristic curves obtained by plotting seeds following the 2D flow  $\mathcal{P}[(c_2\mathbf{l}_1 - c_1\mathbf{l}_2)]$ . (c) shows one possible reconstruction of the characteristic curves. Note how each characteristic curve is reconstructed independently as there is no constraint “across” the curves. Bottom row shows how a successful reconstruction can be achieved when using the proposed shape regularisation scheme with first order regularisation  $\alpha = 0.1, \beta = 0$  (d) and second order regularisation  $\alpha = 0, \beta = 0.5$  (e).



**Fig. 12.5** Shadow segmentation. This experiment shows the result of our shadow region segmentation. From left to right, the three input images (a), (b), (c) and the mask with the resulting shadow labels (d).

### 12.2.5.3 Segmenting Shadowed Regions

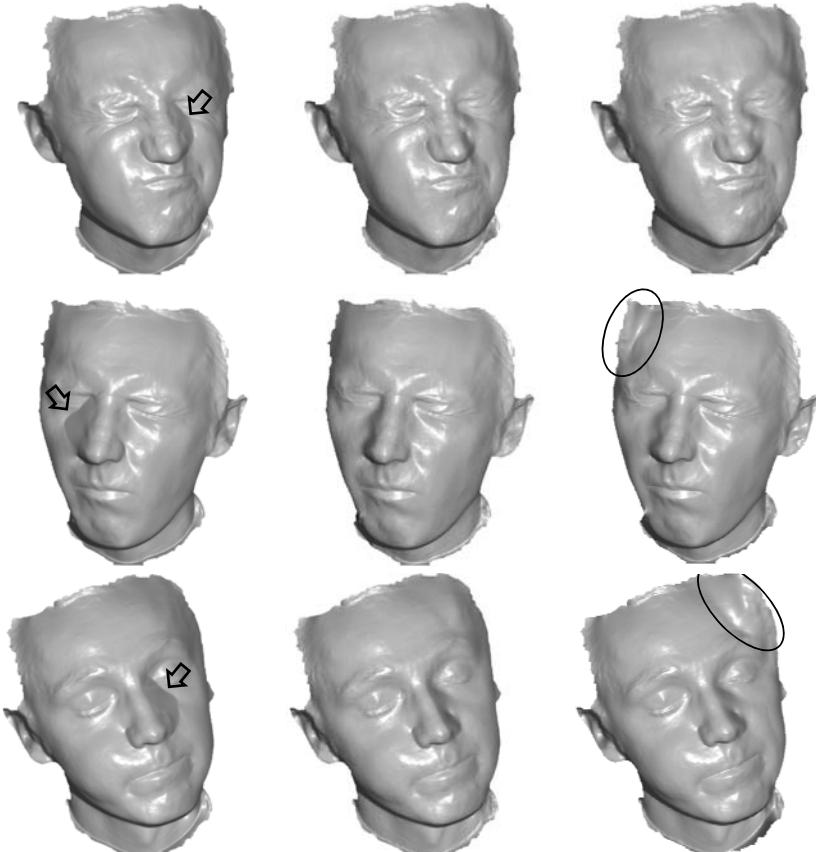
It is known [2] that in photometric stereo with four or more images one can detect shadows by computing the scaled normal that satisfies the constraints in a least squares sense. If the residual of this least squares calculation is high, this implies that the pixel is either in a shadow or in a highlight. With three images however this becomes impossible as the three constraints can always be satisfied exactly, leaving a residual of zero. Recently, [6] proposed a graph-cut based scheme for labelling shadows in photometric stereo with four or more images. Based on the constraint residual, they compute a cost for assigning a particular shadow label to each pixel. This cost is then regularised in an MRF framework where neighbouring pixels are encouraged to have *similar* shadow labels. We would like to use a similar framework but we must supply a different cost for assigning a shadow label. The basic characteristic of a shadow region is that pixel intensities inside it are dark. However this can also occur because of dark surface albedo. To remove the albedo factor we propose to divide pixel intensities with the magnitude of the intensity vector  $\mathbf{c}$ . Our cost for deciding that a pixel is occluded in the  $i$ -th image is  $c_i / \|\mathbf{c}\|$ . This still leaves the possibility that we mistakenly classify a pixel whose normal is nearly perpendicular to the  $i$ -th illumination direction  $\mathbf{l}_i$ . However in that case the pixel is close to being in a self shadow so the risk from misclassifying it is small. The cost for assigning a pixel to the non-shadowed set is given by

$$\frac{1}{\sqrt{3}} - \min_i \frac{c_i}{\|\mathbf{c}\|}.$$

We regularise these costs in an MRF framework under a Potts model pairwise cost [15]. This assigns a fixed penalty for two neighboring pixels being given different shadow labels. The MRF is optimised using the Tree Reweighted message passing algorithm [24]. Figure 12.5 shows an example of applying our shadow region segmentation to a real image.

### 12.2.6 Facial Capture Experiments

We have performed a first experiment with video data of a white-painted face illuminated by three coloured lights in a similar way as in [19]. The setup is calibrated as described in Section 12.2.3. Figure 12.5 shows the three input images obtained from a single colour frame. The automatic shadow segmentation results in Figure 12.5d demonstrate the accuracy of the shadow detection algorithm in Section 12.2.5.3. Figure 12.6 shows three different frames of the video sequence without taking the

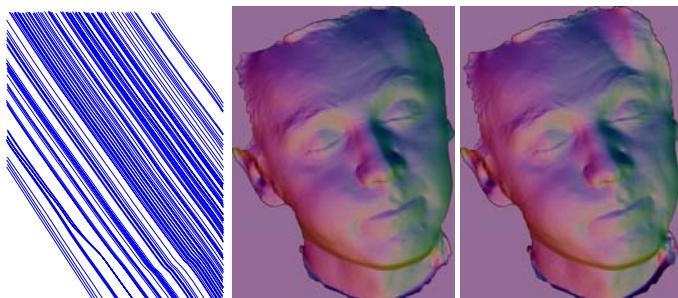


**Fig. 12.6** Face sequence. Three different frames out of a 1000 frame face video sequence. The left column shows the reconstruction when shadows are ignored. Middle and right columns show the corresponding reconstructions after detecting and compensating for the shadow regions using the shading regularisation scheme (middle) and shape regularisation scheme (right). Note the improvement in the regions around the nose reconstruction where strong cast shadows appear (see arrows). Note also how the shape regularisation scheme fails to reconstruct some boundary regions (see circles). This behaviour is further explained in Figure 12.7.

shadows into account (left) and after detecting and adding the shading constraints (middle) and the shape constraints (right). We can appreciate how the nose reconstruction is dramatically improved when correctly processing the shadows (see arrows), even though only two lights are visible in the shadowed regions. We also note that the shape regularisation scheme fails in some boundary regions (see circles in right column) leading to an incorrect reconstruction of the side of the face. This is caused by the Laplacian regularisation term. The term suffers from an ambiguity of two possible solutions, concave or convex, both solutions having similar energy and the data term being unable to disambiguate them.

Figure 12.7 shows a more detailed analysis of the bottom face in figure 12.6. The solution of the shape regularisation scheme agrees with the constraints (Figure 12.7 left) even though it picks the incorrect “concave” solution instead of the convex solution. This is confirmed by looking at the shade rendering of the face under the occluded light (see Figure 12.7 middle and right). The shading regularisation scheme shows a smooth surface (Figure 12.7 middle) while the shape regularisation scheme (Figure 12.7 right) shows a clear artifact. This is expected since the shading regularisation does exactly that, it finds the surface that minimises the variation of the shading when rendering the shape with the occluded light. The extra knowledge of the missing light is exactly what the shape regularisation scheme is missing in order to make the right decision and choose the convex solution.

A second facial performance capture using 19 is shown in figure 12.8. This time the face is not painted, which implies an assumption of constant albedo chromaticity. In order to cope with shadows, the shading regularisation scheme is used. We observe that, despite the constant albedo deviations, *e.g.*, the lips, the system successfully captures fine details such as skin wrinkles.



**Fig. 12.7** Failure case of the shape regularisation scheme. The figures correspond to the bottom face in Figure 12.6. Left shows characteristic curves describing the light occlusion on the right-side of the face. Middle and right show the rendering of the shape under the occluded light using the shading regularisation scheme (middle) and the shape regularisation scheme (right). The failure of the shape regularisation scheme is clearly visible at the top right of the image.



**Fig. 12.8** Face sequence. Acquisition of 3D facial expressions using [19] and the shadow processing technique described in this paper. The shadows are processed with the shading regularisation scheme. The full video sequence has more than a 1000 frames reconstructed.

### 12.2.7 Related Work

The animation and capture of deformations is being explored in many fields, so we provide a general explanation of their relevance in the context of the proposed technique.

#### Texture Cues

White and Forsyth [48, 49] and Scholz *et al* [41] have presented work on using texture cues to perform the specific task of cloth capture. Their methods are based on printing a special pattern on a piece of cloth and capturing video sequences of

that cloth in motion. The estimation of the cloth geometry is based on the observed deformations of the known pattern as well as texture cues extracted from the video sequence. The techniques produce results of very good quality but are ultimately limited by the requirement of printing a special pattern on the cloth which may not be practical for a variety of situations. In the present work, we avoid this requirement while producing detailed results.

Pilet *et al* [38] and Salzmann *et al* [39] proposed a slightly more flexible approach where one uses the pattern already printed in a piece of cloth, by presenting it to the system in a flattened state. Using sparse feature matching the pattern can be detected in each frame of a video sequence. Due to the fact that detection occurs separately in each frame, the method is quite robust to occlusions. However the presented results dealt only with minor non-rigid deformations.

## Photometric Stereo

Photometric stereo [51] is one of the most successful techniques for surface reconstruction from images. It works by observing how changing illumination alters the image intensity of points throughout the object surface. These changes reveal the local surface orientations. This field of local surface orientations can then be integrated into a 3D shape. State of the art photometric-stereo allows uncalibrated light estimation [29, 46] as well as multiple unknown albedos [14, 21]. As mentioned previously, the main difficulty with applying photometric stereo to deforming objects lies in the requirement of changing the light source direction for each captured frame, while the object remains still. This is quite impractical when reconstructing the 3D geometry of a *moving* object. We have shown how multispectral lighting allows one to essentially capture three images (each with a different light direction) in a single snapshot, thus making per-frame photometric reconstruction possible.

## Coloured and Structured Lights

The earliest related works are also the most relevant to the method presented in this Chapter. The first reference to multispectral light for photometric stereo dates back 20 years to the work of Petrov [37]. Ten years later, Kontsevich *et al* [25] actually demonstrated an algorithm for calibrating unknown color light sources and at the same time computing the surface normals of an object in the scene. They verified the theory on synthetic data and an image of a real egg. We use a simplified approach for calibration and the same orientation-from-colour cue to eventually convert video of un-textured cloth into a single surface with complex changing deformations.

More recently, the parameters needed to simulate realistic cloth dynamics were measured in video by projecting explicitly structured horizontal light stripes onto material samples under static and dynamic conditions [5]. This system measured the edges and silhouette mismatches present in real vs. simulated sequences. Many researchers have utilised structured lighting, and Gu *et al* [16] even used colour, although their method is mostly for storing and manipulating acquired surface models of shading and geometry. Weise *et al* [47] is the current state-of-the-art for structured light and has some advantages in terms of absolute 3D depth, but at the expense of

both spatial and temporal sampling, e.g., 17 Hz compared to our 60 Hz (or faster, limited only by the camera used). Zhang *et al* [53] is a nice complete system also with structured lighting that applies to face models and videos. Sand *et al* dispensed with special lighting but leveraged motion capture and automatic silhouettes to deform a human body template [40]. Our technique, on the other hand, expects no prior models of the cloth being reconstructed.

### Shadows in Photometric Stereo

One way of characterising photometric stereo methods is based on the number of different lights required and how they cope with highlights or shadows.

A minimum of 3 lights is required to perform photometric stereo with no extra assumptions [51], and only 2 lights with the additional assumption of constant albedo [35]. Whenever more lights are available, the light visibility problem becomes a labelling problem where each point on the surface has to be assigned to the correct set of lights in order to successfully reconstruct the surface.

For objects with constant albedo, [11] used a Rank-2 constraint to detect surfaces illuminated by only 2 lights. In the case of general albedo, every point on the surface has to be visible in at least 3 images. A 4-light photometric stereo setup was proposed in [34], where light occlusion was detected by checking the consistency of all the possible triplets of lights. The work by [52] was able to detect light occlusions in a 4-light setup and simply treat them as outliers. In [2] a similar algorithm to [34] is presented using a 4-light coloured photometric stereo approach.

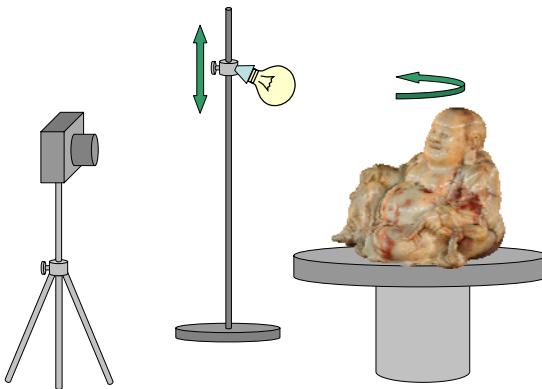
In the recent work by [6], an iterative MRF formulation is proposed for detecting light occlusion and exploiting it as a surface integration constraint. However, the algorithm also requires a minimum of 4 lights and is targeted for setups with a large number of lights.

### 12.3 Multi-view Photometric Stereo

The motivation for the method presented in this Section is digital archiving of 3D objects, a key area of interest in cultural heritage preservation. While laser range scanning is one of the most popular techniques, it has a number of drawbacks, namely the need for specialised, expensive hardware and also the requirement of exclusive access to an object for significant periods of time. Also, for a large class of shiny objects such as porcelain or glazed ceramics, 3D scanning with lasers is challenging [27]. Recovering 3D shape from photographic images is an efficient, cost effective way to generate accurate 3D scans of objects.

Several solutions have been proposed for this long studied problem. When the object is well textured its shape can be obtained by densely matching pixel locations across multiple images and triangulating (see previous Chapter or [42] for a recent review), however the results typically exhibit high frequency noise.

For non textured objects photometric stereo is a well established alternative that can provide very detailed reconstructions. One of the biggest drawbacks



**Fig. 12.9** Our acquisition setup. The object is rotated on a turntable in front of a camera and a point light-source. A sequence of images are captured while the light-source changes position between consecutive frames. No knowledge of the camera or light-source positions is assumed.

of photometric stereo methods is the fact that they can only provide single-viewpoint, 2.5D depth-map reconstructions.

In this Section we describe an elegant and practical method for acquiring a *complete* and *accurate* 3D model from a number of images taken around the object, captured under changing light conditions (see Figure 12.9). The changing (but otherwise unknown) illumination conditions uncover the fine geometric detail of the object surface which is obtained by a generalised photometric stereo scheme.

The object's reflectance is assumed to follow Lambert's law, *i.e.*, points on the surface keep their appearance constant irrespective of viewpoint. The method can however tolerate isolated specular highlights, typically observed in glazed surfaces such as porcelain. We also assume that a single, distant light-source illuminates the object and that it can be changed arbitrarily between image captures. Finally, it is assumed that the object can be segmented from the background and silhouettes extracted automatically.

### 12.3.1 Related Work

The method presented here draws inspiration from the recent work of [29] where the authors explore the possibility of using photometric stereo with images from multiple views, when correspondence between views is not initially known. Picking an arbitrary viewpoint as a reference image, a depth-map with respect to that view serves as the source of approximate correspondences between frames. This depth-map is initialised from a Delaunay triangulation of sparse 3D features located on the surface. Using this depth-map, their algorithm performs a photometric stereo computation obtaining normal directions for each depth-map location. When these normals are integrated, the resulting depth-map is closer to the true surface than

the original. The paper presents high quality reconstructions and gives a theoretical argument justifying the convergence of the scheme. The method however relies on the existence of distinct features on the object surface which are tracked to obtain camera motion and initialise the depth-map. In the class of textureless objects we are considering, it may be impossible to locate such surface features and indeed our method has no such requirement. Also the surface representation is still depth-map based and consequently the models produced are 2.5D.

A similar approach of extending photometric stereo to multiple views and more complex BRDFs was presented in [36] with the limitation of almost planar 2.5D reconstructed surfaces. Our method is based on the same fundamental principle of bootstrapping photometric stereo with approximate correspondences, but we use a general volumetric framework which allows complete 3D reconstructions from multiple views.

Quite related to this idea is the work of [3] and [33] where photometric stereo information is combined with 3D range scan data. In [3] the photometric information is simply used as a normal map texture for visualisation purposes. In [33], a very good initial approximation to the object surface is obtained using range scanning technology, which however is shown to suffer from high-frequency noise. By applying a fully calibrated 2.5D photometric stereo technique, normal maps are estimated which are then integrated to produce an improved, almost noiseless surface geometry. Our acquisition technique is different from [33] in the following respects: (1) we only use standard photographic images and simple light sources, (2) our method is fully uncalibrated- all necessary information is extracted from the object's contours and (3) we completely avoid the time consuming and error prone process of merging 2.5D range scans.

The use of the silhouette cue is inspired by the work of [45] where a scheme for the recovery of illumination information, surface reflectance and geometry is described. The algorithm described makes use of frontier points, a geometrical feature of the object obtained by the silhouettes. Frontier points are points of the visual hull where two contour generators intersect and hence are guaranteed to be on the object surface. Furthermore the local surface orientation is known at these points, which makes them suitable for various photometric computations such as extraction of reflectance and illumination information. Our method generalises the idea by examining a much richer superset of frontier points which is the set of contour generator points. We overcome the difficulty of localising contour generators by a robust random sampling strategy. The price we pay is that a considerably simpler reflectance model must be used.

Although solving a different type of problem, the work of [23] is also highly related mainly because the class of objects addressed is similar to ours. While the energy term defined and optimised in their paper bears strong similarity to ours, their reconstruction setup keeps the lights fixed with respect to the object so in fact an entirely different problem is solved and hence a performance comparison between the two techniques is difficult. However the results presented in [23] at first glance seem to be lacking in detail especially in concavities, while our technique considerably improves on the visual hull. Finally, there is a growing volume of work on

using specularities for calibrating photometric stereo (see [10] for a detailed literature survey). This is an example of a different cue used for performing uncalibrated photometric stereo on objects of the same class as the one considered here. However methods proposed have so far only been concerned with the fixed view case.

### 12.3.2 Algorithm

The method presented here reconstructs the complete geometry of 3D objects by exploiting the powerful silhouette and shading cues. We modify classic photometric stereo and cast it in a multi-view framework where the camera is allowed to circumnavigate the object and illumination is allowed to vary. Firstly, the object's silhouettes are used to recover camera motion using the technique presented in [18], and via a novel robust estimation scheme they allow us to accurately estimate the light directions and intensities in every image.

Secondly, the object surface, which is parametrised by a mesh and initialised from the visual hull, is evolved until its predicted appearance matches the captured images. The advantages of our approach are the following:

- It is fully uncalibrated: no light or camera pose calibration object needs to be present in the scene. Both camera pose and illumination are estimated from the object's silhouettes.
- The full 3D geometry of a complex, textureless multi-albedo object is accurately recovered, something not previously possible by any other method.
- It is practical and efficient as evidenced by our simple acquisition setup.

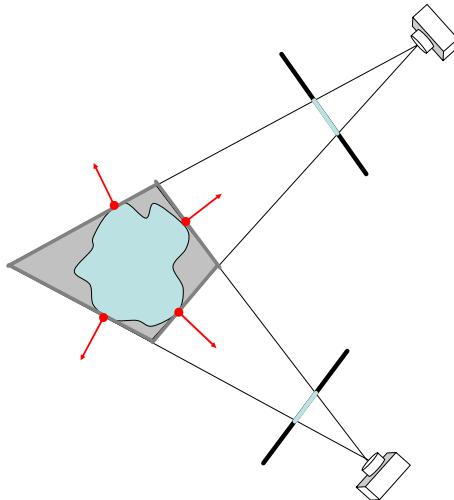
#### 12.3.2.1 Robust Estimation of Light-Sources from the Visual Hull

For an image of a lambertian object with varying albedo, under a single distant light source, and assuming no self-occlusion, each surface point projects to a point of intensity given by:

$$c = \mathbf{l}^T \rho \mathbf{n}, \quad (12.15)$$

where  $\mathbf{l}$  is a 3D vector directed towards the light-source and scaled by the light-source intensity,  $\mathbf{n}$  is the surface unit normal at the object location and  $\rho$  is the albedo at that location. Equation (12.15) provides a single constraint on the three coordinates of the product  $\rho \mathbf{l}$ . Then, given three points  $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$  with an unknown but *equal* albedo  $\rho$ , their normals (non co-planar)  $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3$ , and the corresponding three image intensities  $i_1, i_2, i_3$ , we can construct three such Equations that can uniquely determine  $\rho \mathbf{l}$  as

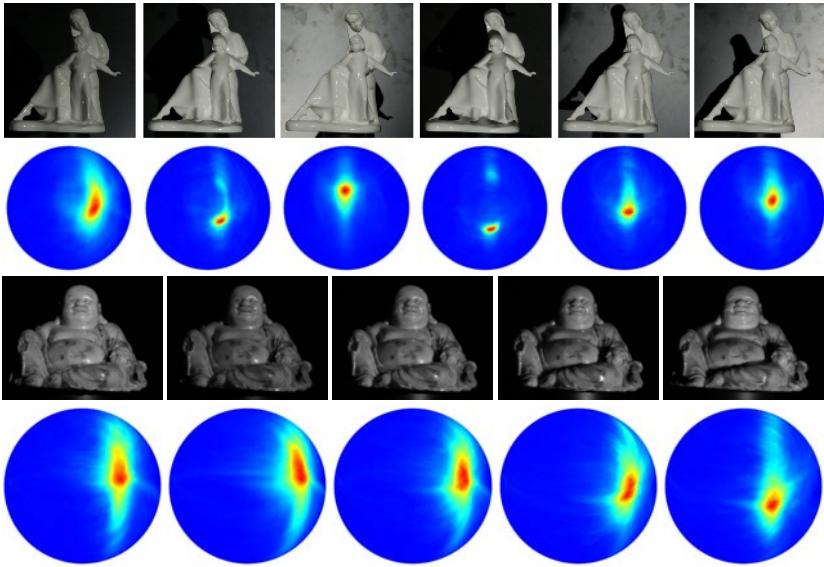
$$\rho \mathbf{l} = [\mathbf{n}_1 \mathbf{n}_2 \mathbf{n}_3]^{-1} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}. \quad (12.16)$$



**Fig. 12.10** The visual hull for light estimation. The figure shows a 2D example of an object which is photographed from two viewpoints. The visual hull (gray quadrilateral) is the largest volume that projects inside the silhouettes of the object. While the surface of the visual hull is generally quite far from the true object surface, there is a set of points where the two surfaces are tangent and moreover, share the same local orientation (these points are denoted here with the four dots and arrows). In the full 3D case, three points with their surface normals, are enough to fix an illumination hypothesis, against which all other points can be tested for agreement. This suggests a robust random sampling scheme, described in the main text, via which the correct illumination can be obtained.

For multiple images, these same three points can provide the light directions and intensities in each image up to a global unknown scale factor  $\rho$ . The problem is then how to obtain three such points.

Our approach is to use the powerful silhouette cue. The observation on which this is based is the following: when the images have been calibrated for camera motion, the object's silhouettes allow the construction of the *visual hull* [26], which is defined as the maximal volume that projects inside the silhouettes (see Figure 12.10). A fundamental property of the visual hull is that its surface coincides with the real surface of the object along a set of 3D curves, one for each silhouette, known as *contour generators* [8]. Furthermore, for all points on those curves, the surface orientation of the visual hull surface is equal to the orientation of the object surface. Therefore if we could detect points on the visual hull that belong to contour generators and have equal albedo, we could use their surface normal directions and projected intensities to estimate lighting. Unfortunately contour generator points with equal albedo cannot be directly identified within the set of all points of the visual hull. Light estimation however can be viewed as robust model fitting where the inliers are the contour generator points of some constant albedo and the outliers are the rest of the visual hull points. The albedo of the inliers will be the *dominant* albedo,



**Fig. 12.11** Shape of illumination consensus. For different illumination configurations we have plotted the consensus as a function of light direction. For each direction consensus has been maximised with respect to light intensity. Red values denote big consensus. The shape of the maxima of this cost function as well as the lack of local optima implies a stable optimisation problem. Top: 6 different illuminations of a single albedo object. Bottom: 4 different illuminations of a multi-albedo object. Although the presence of multiple albedos degrades the quality of the light estimation (the peak is broader), it is still a clear single optimum.

i.e., the colour of the majority of the contour generator points. One can expect that the outliers do not generate consensus in favour of any particular illumination model while the inliers do so in favour of the correct model. This observation motivates us to use a robust RANSAC scheme [13] to separate inliers from outliers and estimate illumination direction and intensity. The scheme can be summarised as follows:

1. Pick three points on the visual hull and from their image intensities and normals estimate an illumination hypothesis for  $\rho\mathbf{l}$ .
2. Every point on the visual hull  $\mathbf{x}_m$  will now vote for this hypothesis if its predicted image intensity is within a given threshold  $\tau$  of the observed image intensity  $c_m$ , i.e.,

$$|\rho\mathbf{l}^T \cdot \mathbf{n}_m - c_m| < \tau, \quad (12.17)$$

where  $\tau$  allows for quantisation errors, image noise, etc.

3. Repeat 1 and 2 a set number of times always keeping the illumination hypothesis with the largest number of votes.

The shape of the actual function being optimised by the RANSAC scheme described above was explored graphically for a porcelain object in Figure 12.11. The number of points voting for a light direction (maximised with respect to light intensity) was plotted as a 2D function of latitude and longitude of the light direction. These graphical representations, obtained for six different illuminations, show the lack of local optima and the presence of clearly defined maxima.

This simple method can also be extended in the case where the illumination is kept fixed with respect to the camera for  $K$  frames. This corresponds to  $K$  illumination vectors  $R_1\mathbf{l}, \dots, R_K\mathbf{l}$  where  $R_k$  are  $3 \times 3$  rotation matrices that rotate the fixed illumination vector  $\mathbf{l}$  with respect to the object. In that case a point on the visual hull  $\mathbf{x}_m$  with normal  $\mathbf{n}_m$  will vote for  $\mathbf{l}$  if it is visible in the  $k$ -th image where its intensity is  $c_{m,k}$  and

$$|\rho(R_k\mathbf{l})^T \cdot \mathbf{n}_m - c_{m,k}| < \tau. \quad (12.18)$$

A point is allowed to vote more than once if it is visible in more than one image.

Even though in theory the single image case suffices for independently recovering illumination in each image, in our acquisition setup light can be kept fixed over more than one frame. This allows us to use the extended scheme in order to further improve our estimates. A performance comparison between the single view and the multiple view case is provided through simulations with synthetic data in the experiments Section.

An interesting and very useful byproduct of the robust RANSAC scheme is that any deviations from our assumptions of a Lambertian surface of uniform albedo are rejected as outliers. This provides the light estimation algorithm with a degree of tolerance to sources of error such as highlights or local albedo variations. The next Section describes the second part of the algorithm which uses the estimated illumination directions and intensities to recover the object surface.

### 12.3.2.2 Fusing Multiple Views

Having estimated the distant light-source directions and intensities for each image our goal is to find a closed 3D surface that is photometrically consistent with the images and the estimated illumination, *i.e.*, its predicted appearance by the lambertian model and the estimated illumination matches the images captured. To achieve this we use an optimisation approach where a cost function penalising the discrepancy between images and predicted appearance is minimised.

Our algorithm optimises a surface  $S$  that is represented as a mesh with vertices  $\mathbf{x}_1, \dots, \mathbf{x}_M$ , triangular faces  $f = 1, \dots, F$  and corresponding albedo  $\rho_1, \dots, \rho_F$ . We denote by  $\mathbf{n}_f$  and  $A_f$  the mesh normal and the surface area at face  $f$ . Also let  $c_{f,k}$  be the intensity of face  $f$  on image  $k$  and let the set  $\mathcal{V}_f$  be the set of images (subset of  $\{1, \dots, K\}$ ) from which face  $f$  is visible. The light direction and light intensity of the  $k$ -th image will be denoted by  $\mathbf{l}_k$ .

We use a scheme similar to the ones used in [23, 46] where the authors introduce a decoupling between the mesh normals  $\mathbf{n}_1 \dots \mathbf{n}_F$ , and the direction vectors used in the Lambertian model Equation. We call these new direction vectors  $\mathbf{v}_1 \dots \mathbf{v}_F$

*photometric normals*, and they are independent of the mesh normals. The minimisation cost is then composed of two terms, where the first term  $E_v$  links the photometric normals to the observed image intensities:

$$E_v(\mathbf{v}_{1,\dots,F}, \rho_{1,\dots,F}; \mathbf{x}_{1,\dots,M}) = \sum_{f=1}^F \sum_{k \in \mathcal{V}_f} (\mathbf{l}_k^T \rho_f \mathbf{v}_f - c_{f,k})^2, \quad (12.19)$$

and the second term  $E_m$  brings the mesh normals close to the photometric normals through the following Equation:

$$E_m(\mathbf{x}_{1,\dots,M}; \mathbf{v}_{1,\dots,F}) = \sum_{f=1}^F \|\mathbf{n}_f - \mathbf{v}_f\|^2 A_f. \quad (12.20)$$

This decoupled energy function is optimised by iterating the following two steps:

1. **Photometric normal optimisation.** The vertex locations are kept fixed while  $E_v$  is optimised with respect to the photometric normals and albedos. This is achieved by solving the following independent minimisation problems for each face  $f$ :

$$\mathbf{v}_f, \rho_f = \arg \min_{\mathbf{v}, \rho} \sum_{k \in \mathcal{V}_f} (\mathbf{l}_k^T \rho \mathbf{v} - c_{f,k})^2 \text{ s.t. } \|\mathbf{v}\| = 1. \quad (12.21)$$

2. **Vertex optimisation.** The photometric normals are kept fixed while  $E_m$  is optimised with respect to the vertex locations using gradient descent.

These two steps are interleaved until convergence which takes about 20 steps for the sequences we experimented with. Typically each integration phase takes about 100 gradient descent iterations. Note that for the first step described above, *i.e.*, evolving the mesh until the surface normals converge to some set of *target* orientations, a variety of solutions is possible. A slightly different solution to the same geometric optimisation problem has recently been proposed in [33], where the target orientations are assigned to each vertex, rather than each face as we do here. That formulation lends itself to a closed-form solution with respect to the position of a single vertex. An iteration of these local vertex displacements yields the desired convergence. As both formulations offer similar performance, the choice between them should be made depending on whether the target orientations are given on a per vertex or per facet basis.

The visibility map  $\mathcal{V}_f$  is a set of images in which we can measure the intensity of face  $f$ . It excludes images in which face  $f$  is occluded using the current surface estimate as the occluding volume as well as images where face  $f$  lies in shadow. Shadows are detected by a simple thresholding mechanism, *i.e.*, face  $f$  is assumed to be in shadow in image  $k$  if  $c_{f,k} < \tau_{\text{shadow}}$  where  $\tau_{\text{shadow}}$  is a sufficiently low intensity threshold. Due to the inclusion of a significant number of viewpoints in  $\mathcal{V}_f$ , (normally at least 4) the system is quite robust to the choice of  $\tau_{\text{shadow}}$ . For all the experiments presented here, the value  $\tau_{\text{shadow}} = 5$  was used (for intensities in the range 0-255). As for the highlights, we also define a threshold  $\tau_{\text{highlight}}$  such as a face  $f$  is assumed to be on a highlight in image  $k$  if  $c_{f,k} > \tau_{\text{highlight}}$ . In order to compute

```

Capture images of object.
Extract silhouettes.
Recover camera motion and compute visual hull.
Estimate light directions and intensities in every image (Section 3.2.1).
Initialise a mesh with vertices  $x_1 \dots x_M$  and faces  $f = 1 \dots F$  to the object's visual hull.
while mesh-not-converged do
    Optimise  $E_v$  with respect to  $v_1 \dots v_F$  (19).
    Optimise  $E_m$  with respect to  $x_1 \dots x_M$  (20).
end while

```

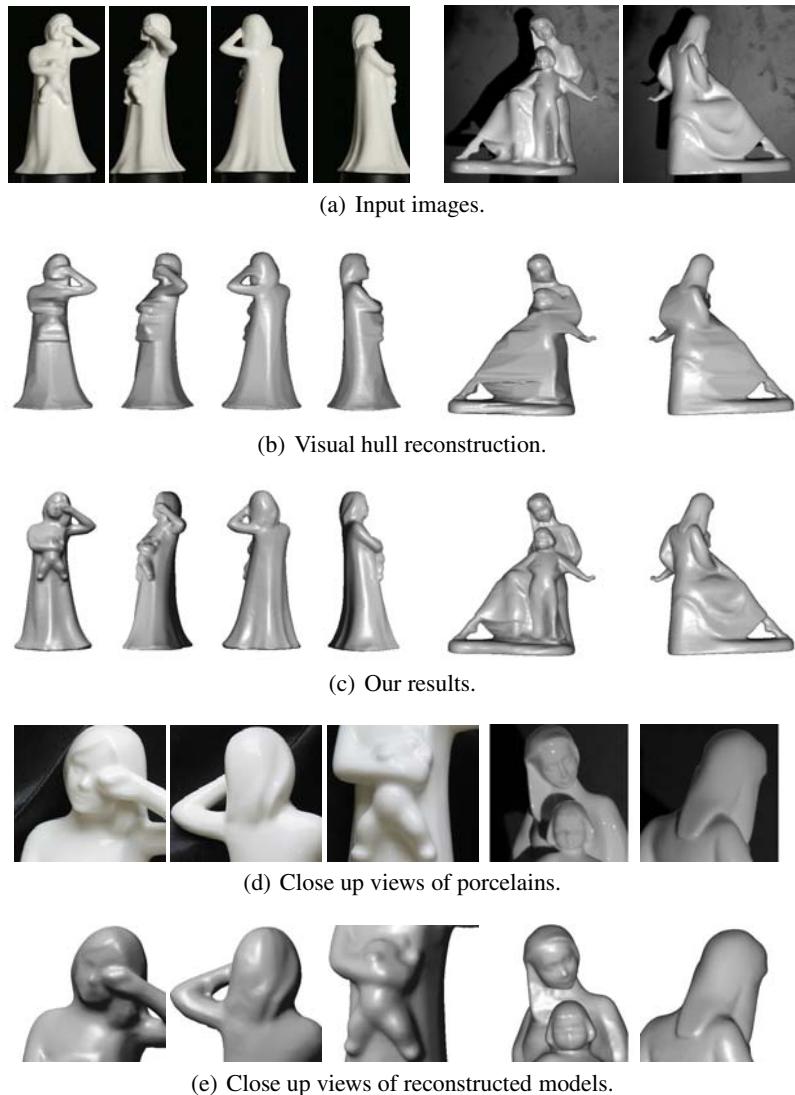
**Fig. 12.12** The multi-view reconstruction algorithm.

$\tau_{highlight}$  need to distinguish between single albedo objects and multi-albedo objects. Single albedo objects are easily handled since the light calibration step gives us the light intensity. Hence, under the Lambertian assumption, no point on the surface can produce an intensity higher than the light intensity, *i.e.*,  $\tau_{highlight} = ||\rho l||$ . In the multi-albedo case  $\rho$  can also vary, and it is likely that the albedo picked by the robust light estimation algorithm is not the brightest one present on the object. As a result, we prefer to use a global threshold to segment the highlights on the images. It is worth noting that this approach works for the porcelain objects because highlights are very strong and localised, so just a simple sensor saturation test is enough to find them, *i.e.*,  $\tau_{highlight} = 254$ .

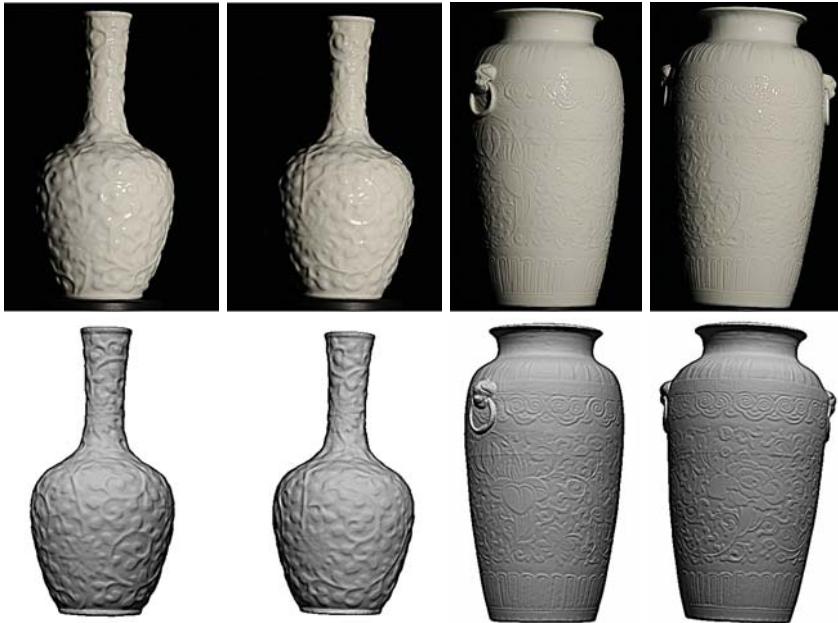
### 12.3.3 Experiments

The setup used to acquire the 3D model of the object is quite simple (see Figure 12.9). It consists of a turntable, onto which the object is mounted, a 60W halogen lamp and a digital camera. The object rotates on the turntable and 36 images (*i.e.*, a constant angle step of 10 degrees) of the object are captured by the camera while the position of the lamp is changed. In our experiments we have used three different light positions which means that the position of the lamp was changed after twelve, and again after twenty-four frames. The distant light source assumptions are satisfied if an object of 15cm extent is placed 3-4m away from the light.

The algorithm was tested on five challenging shiny objects, two porcelain figurines shown in Figure 12.13, two fine relief Chinese Qing-dynasty porcelain vases shown in Figure 12.14, and one textured Jade Buddha figurine in Figure 12.15. Thirty-six  $3456 \times 2304$  images of each of the objects were captured under three different illuminations. The object silhouettes were extracted by intensity thresholding and were used to estimate camera motion and construct the visual hull (second row of Figure 12.13). The visual hull was processed by the robust light estimation scheme of Section 12.3.2.1 to recover the distance light-source directions and intensities in each image. The photometric stereo scheme of Section 12.3.2.2 was then applied. The results in Figure 12.14 show reconstructions of porcelain vases with very fine relief. The reconstructed relief (especially for the vase on the right) is less than a millimetre while their height is approximately 15-20 cm. Figure 12.15



**Fig. 12.13** Reconstructing porcelain figurines. Two porcelain figurines reconstructed from a sequence of 36 images each (some of the input images are shown in (a)). The object moves in front of the camera and illumination (a 60W halogen lamp) changes direction twice during the image capture process. (b) shows the results of a visual hull reconstruction while (c) shows the results of our algorithm. (d) and (e) show detailed views of the figurines and the reconstructed models respectively.



**Fig. 12.14** Reconstructing Chinese Qing-dynasty porcelain vases. Top: sample of input images. Bottom: proposed method. The resulting surface captures all the fine details present in the images, even in the presence of strong highlights.

shows a detailed reconstruction of a Buddha figurine made of polished Jade. This object is actually textured, which implies classic stereo algorithms could be applied. Using the camera motion information and the captured images, a state-of-the-art multi-view stereo algorithm [17] was executed. The results are shown in the second row of Figure 12.15. It is evident that, while the low frequency component of the geometry of the figurine is correctly recovered, the high frequency detail obtained by [17] is noisy. The reconstructed model appears bumpy even though the actual object is quite smooth. Our results do not exhibit surface noise while capturing very fine details such as surface cracks.

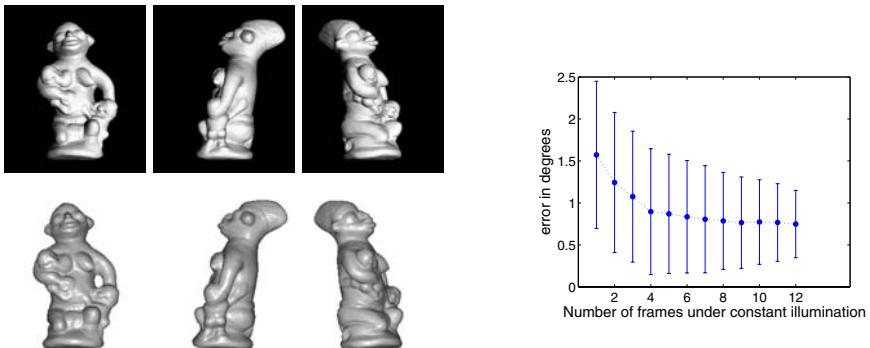
To quantitatively analyse the performance of the multi-view photometric stereo scheme presented here with ground truth, an experiment on a synthetic scene was performed (Figure 12.16). A 3D model of a sculpture (digitised via a different technique) was rendered from 36 viewpoints with uniform albedo and using the Lambertian reflectance model. The 36 frames were split into three sets of 12 and within each set the single distant illumination source was held constant. Silhouettes were extracted from the images and the visual hull was constructed. This was then used to estimate the illumination direction and intensity as described in Section 12.3.2.1. In 1000 runs of the illumination estimation method for the synthetic scene, the mean light direction estimate was 0.75 degrees away from the true direction with a standard deviation of 0.41 degrees. The model obtained by our algorithm was compared



**Fig. 12.15** Reconstructing coloured jade. Left: Two input images. Middle: model obtained by multi-view stereo method from [17]. Right: proposed method. The resulting surface is filtered from noise while new high frequency geometry is revealed (note the reconstructed surface cracks in the middle of the figurine's back).

to the ground truth surface by measuring the distance of each point on our model from the closest point in the ground truth model. This distance was found to be about 0.5mm when the length of the biggest diagonal of the bounding box volume was defined to be 1m. Even though this result was obtained from perfect noiseless images it is quite significant since it implies that any loss of accuracy can only be attributed to the violations of our assumptions rather than the optimisation methods themselves. Many traditional multi-view stereo methods would not be able to achieve this due to the strong regularisation that must be imposed on the surface. By contrast our method requires no regularisation when faced with perfect noiseless images.

Finally, we investigated the effect of the number of frames during which illumination is held constant with respect to the camera frame. Our algorithm can in theory obtain the illumination direction and intensity in every image independently. However keeping the lighting fixed over two or more frames, and supplying that knowledge to the algorithm can significantly improve estimates. The next experiment was designed to test this improvement by performing a light estimation over  $K$  images where the light has been kept fixed with respect to the camera. The results are plotted in Figure 12.16 right and show the improvement of the accuracy of the recovered lighting directions as  $K$  increases from 1 to 12. The metric used was the angle between the ground truth light direction and the estimated light direction over 1000 runs of the robust estimation scheme. For  $K = 1$  the algorithm achieves a mean error of 1.57 degrees with a standard deviation of 0.88 while for  $K = 12$  it



**Fig. 12.16** Synthetic evaluation. Left: the accuracy of the algorithm was evaluated using an image sequence synthetically generated from a 3D computer model of a sculpture. This allowed us to compare the quality of the reconstructed model against the original 3D model as well as measure the accuracy of the light estimation. The figure shows the reconstruction results obtained, below the images of the synthetic object. The mean distance of all points of the reconstructed model from the ground truth was found to be about 0.5mm if the bounding volume’s diagonal is 1m. Right: The figure shows the effect of varying the length of the frame subsequences that have constant light. The angle between the recovered light direction and ground truth has been measured for 1000 runs of the RANSAC scheme for each number of frames under constant lighting. With just a single frame per illumination the algorithm achieves a mean error of 1.57 degrees with a standard deviation of 0.88 degrees. With 12 frames sharing the same illumination the mean error drops to 0.75 degrees with a standard deviation of 0.41 degrees.

achieves 0.75 degrees with a standard deviation of 0.41 degrees. The decision for selecting a value for  $K$  should be a consideration of the tradeoff between practicality and maximising the total number of different illuminations in the sequence which is  $M/K$  where  $M$  is the total number of frames.

## References

1. Amit, G.F., Agrawal, A., Raskar, R.: What is the range of surface reconstructions from a gradient field? In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 578–591. Springer, Heidelberg (2006)
2. Barsky, S., Petrou, M.: The 4-source photometric stereo technique for three-dimensional surfaces in the presence of highlights and shadows. IEEE Transaction on Pattern Analysis and Machine Intelligence 25(10), 1239–1252 (2003)
3. Bernardini, F., Rushmeier, H., Martin, I., Mittleman, J., Taubin, G.: Building a digital model of michelangelo’s florentine pieta. IEEE Computer Graphics and Applications 22(1), 59–67 (2002)
4. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. SIGGRAPH, 417–424 (2000)

5. Bhat, K.S., Twigg, C.D., Hodgins, J.K., Khosla, P.K., Popović, Z., Seitz, S.M.: Estimating cloth simulation parameters from video. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer animation, pp. 37–51 (2003)
6. Chandraker, M., Agarwal, S., Kriegman, D.: Shadowcuts: Photometric stereo with shadows. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
7. Chen, H., Belhumeur, P., Jacobs, D.: In search of illumination invariants. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 254–261 (2000)
8. Cipolla, R., Giblin, P.: Visual Motion of curves and surfaces. Cambridge University Press, Cambridge (1999)
9. Davis, T.A.: Algorithm 832: Umfpack, an unsymmetric-pattern multifrontal method. ACM Transactions on Mathematical Software 30(2), 196–199 (2004)
10. Dbrohlav, O., Chandler, M.: Can two specular pixels calibrate photometric stereo? In: Proceedings of the International Conference on Computer Vision (2005)
11. Drew, M.: Reduction of rank-reduced orientation-from-color problem with many unknown lights to two-image known-illuminant photometric stereo. In: Proceedings of the International Symposium on Computer Vision, pp. 419–424 (1995)
12. Fan, J., Wolff, L.B.: Surface curvature and shape reconstruction from unknown multiple illumination and integrability. Computer Vision and Image Understanding 65(2), 347–359 (1997)
13. Fischler, M.A., Bolles, R.C.: Ransac, random sampling consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 26, 381–395 (1981)
14. Goldman, D.B., Curless, B., Hertzmann, A., Seitz, S.M.: Shape and spatially-varying BRDFs from photometric stereo. In: Proceedings of the International Conference on Computer Vision, pp. 341–348 (2005)
15. Greig, D., Porteous, B., Seheult, A.: Exact maximum a posteriori estimation for binary images. Journal of the Royal Statistical Society 51(2), 271–279 (1989)
16. Gu, X., Zhang, S., Huang, P., Zhang, L., Yau, S.T., Martin, R.: Holoimages. In: Proceedings of the ACM Symposium on Solid and Physical Modeling, pp. 129–138 (2006)
17. Hernández, C., Schmitt, F.: Silhouette and stereo fusion for 3d object modeling. Computer Vision and Image Understanding 96(3), 367–392 (2004)
18. Hernández, C., Schmitt, F., Cipolla, R.: Silhouette coherence for camera calibration under circular motion. IEEE Transaction on Pattern Analysis and Machine Intelligence 29(2), 343–349 (2007)
19. Hernández, C., Vogiatzis, G., Brostow, G., Stenger, B., Cipolla, R.: Non-rigid photometric stereo with colored lights. In: Proceedings of the International Conference on Computer Vision (2007)
20. Hertzmann, A., Seitz, S.: Shape and materials by example: a photometric stereo approach. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 533–540 (2003)
21. Hertzmann, A., Seitz, S.: Shape reconstruction with general, varying brdfs. IEEE Transaction on Pattern Analysis and Machine Intelligence 27(8), 1254–1264 (2005)
22. Horn, B.K.P.: Robot vision. MIT Press, Cambridge (1986)
23. Jin, H., Cremers, D., Yezzi, A., Soatto, S.: Shedding light in stereoscopic segmentation. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 36–42 (2004)
24. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. IEEE Transaction on Pattern Analysis and Machine Intelligence 28(10), 1568–1583 (2006)

25. Kontsevich, L., Petrov, A., Vergelskaya, I.: Reconstruction of shape from shading in color images. *Journal of the Optical Society of America A* 11(3), 1047–1052 (1994)
26. Laurentini, A.: The visual hull concept for silhouette-based image understanding. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 16(2) (1994)
27. Levoy, M.: Why is 3d scanning hard? In: Invited address at 3D Processing, Visualization, Transmission, Padua, Italy (2002)
28. Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J., Fulk, D.: The digital michelangelo project: 3d scanning of large statues. In: Proceedings of the ACM SIGGRAPH, pp. 15–22 (2000)
29. Lim, J., Ho, J., Yang, M.H., Kriegman, D.: Passive photometric stereo from motion. In: Proceedings of the International Conference on Computer Vision, pp. 1635–1642 (2005)
30. Lin, S., Lee, S.: Estimation of diffuse and specular appearance. In: Proceedings of the International Conference on Computer Vision, pp. 855–860 (1999)
31. Malzbender, T., Wilburn, B., Gelb, D., Ambrisco, B.: Surface enhancement using real-time photometric stereo and reflectance transformation. In: Proceedings of the Eurographics Symposium on Rendering (2006)
32. Nayar, S., Ikeuchi, K., Kanade, T.: Surface reflection: physical and geometrical perspectives. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 13(7), 611–634 (1991)
33. Nehab, D., Rusinkiewicz, S., Davis, J., Ramamoorthi, R.: Efficiently combining positions and normals for precise 3d geometry. In: Proceedings of the ACM SIGGRAPH, pp. 536–543 (2005)
34. North Coleman Jr., E., Jain, R.: Obtaining 3-dimensional shape of textured and specular surfaces using four-source photometry. In: Shape recovery, pp. 180–199. Jones and Bartlett Publishers, Inc., USA (1992)
35. Onn, R., Bruckstein, A.: Integrability disambiguates surface recovery in two-image photometric stereo. *International Journal of Computer Vision* 5(1), 105–113 (1990)
36. Paterson, J., Claus, D., Fitzgibbon, A.: Brdf and geometry capture from extended inhomogeneous samples using flash photography. In: Proceedings of Eurographics (2005)
37. Petrov, A.: Light, color and shape. *Cognitive Processes and their Simulation* (in Russian), 350–358 (1987)
38. Pilet, J., Lepetit, V., Fua, P.: Real-time non-rigid surface detection. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2005)
39. Salzmann, M., Ilic, S., Fua, P.: Physically valid shape parameterization for monocular 3-d deformable surface tracking. In: Proceedings of the British Machine Vision Conference (2005)
40. Sand, P., McMillan, L., Popović, J.: Continuous capture of skin deformation. *ACM Transaction on Graphics* 22(3), 578–586 (2003)
41. Scholz, V., Stich, T., Keckeisen, M., Wacker, M., Magnor, M.: Garment motion capture using color-coded patterns. *Computer Graphics Forum* 24(3), 439–448 (2005)
42. Seitz, S., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 519–528 (2006)
43. Tankus, A., Kiryati, N.: Photometric stereo under perspective projection. In: Proceedings of the International Conference on Computer Vision (2005)
44. Treuille, A., Hertzmann, A., Seitz, S.: Example-based stereo with general brdfs. In: Pandila, T., Matas, J(G.) (eds.) *ECCV 2004. LNCS*, vol. 3022, pp. 457–469. Springer, Heidelberg (2004)

45. Vogiatzis, G., Favaro, P., Cipolla, R.: Using frontier points to recover shape, reflectance and illumination. In: Proceedings of the International Conference on Computer Vision, pp. 228–235 (2005)
46. Vogiatzis, G., Hernández, C., Cipolla, R.: Reconstruction in the round using photometric normals and silhouettes. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 1847–1854 (2006)
47. Weise, T., Leibe, B., Gool, L.V.: Fast 3d scanning with automatic motion compensation. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
48. White, R., Forsyth, D.: Combining cues: Shape from shading and texture. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 1809–1816 (2006)
49. White, R., Forsyth, D.: Retexturing single views using texture and shading. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 70–81. Springer, Heidelberg (2006)
50. Wolff, L.B., Angelopoulou, E.: 3d stereo using photometric ratios. In: Eklundh, J.-O. (ed.) ECCV 1994. LNCS, vol. 801, pp. 247–258. Springer, Heidelberg (1994)
51. Woodham, R.: Photometric method for determining surface orientation from multiple images. Optical Engineering 19(1), 139–144 (1980)
52. Yuille, A., Snow, D.: Shape and albedo from multiple images using integrability. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (1997)
53. Zhang, L., Snavely, N., Curless, B., Seitz, S.M.: Spacetime faces: High-resolution capture for modeling and animation. In: Proceedings of ACM Annual Conference on Computer Graphics, pp. 548–558 (2004)

# Index

- 3D hand tracking, 235, 237  
3D object, 207, 209, 212, 216, 218, 222, 281, 330, 333  
3D object category, 209, 211, 213, 216  
3D shape, 281, 282, 329, 330  
3D tracking, 238
- action recognition, 206  
activity recognition, 157  
AdaBoost, 237, 239, 243  
affine motion, 46  
alignment, 46, 47  
aspect graphs, 207
- bag of codewords, 160, 167, 173, 174, 183, 187, 188, 191, 200, 214, 227  
belief propagation, 76, 93, 95, 120, 291  
bidirectional reflectance distribution function, 13  
BRDF, 13, 332
- calibration, 281, 285, 292, 307, 313, 315, 317, 329, 333, 338  
canonical correlation analysis, 135  
canonical parts, 210, 212, 214, 216, 219, 221, 222, 224  
canonical view, 212, 214, 217, 219, 221, 222  
cast shadow, 13  
CCA, 135  
character recognition, 144  
clique, 53, 114, 118, 121, 133, 290, 295
- clustering, 173–176, 200, 275–278  
codebook, 162, 163, 165, 166, 173, 175, 200  
collaborative tracking, 271, 272  
Computer Vision, 51–55, 57–59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99, 101, 103, 105, 107, 109, 112, 122, 123, 133, 158, 182, 183, 205, 227, 239, 281  
conditional random fields, 52, 53, 70, 74, 75, 109, 113, 114, 199  
constraint satisfaction, 53  
context-based classification, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129, 131, 133  
correspondence, 282–284, 331, 332  
CRF, 53, 59, 67, 70, 74, 80, 82, 84, 85, 91, 92, 109, 113, 115, 117, 119, 121, 133
- defocus, 282  
detection, 30, 77, 82–84, 91, 92, 109, 111, 112, 115, 122–125, 127–129, 131, 133, 160, 200, 205, 210, 212, 222–224, 227, 236, 238, 241–243, 245, 247, 249, 251, 254, 263, 267, 269, 271, 279, 313, 326, 329  
detection-based tracking, 265, 273, 276  
discriminative graphical models, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129, 131, 133  
discriminative model, 109, 113, 116, 118, 123, 129, 160, 190  
disparity estimation, 51  
dynamic algorithms, 58, 71, 72

- EER, 147  
 energy minimization, 53 54 56 58 84 93 94  
 ensemble tracking, 239  
 equal error rate, 147  
 event recognition, 159 160 166  
 exact minimization, 51
- face recognition, 135 139 147 148 154  
 false acceptance rate, 147  
 false rejection rate, 147  
 FAR, 147 150 151 154  
 feature descriptor, 28  
 feature detector, 28 44  
 feature matching, 28 29 40  
 features evaluation, 28 29 34  
 Frobenius matrix, 37  
 FRR, 147 150 151 154
- Gauss-Hermite kernel, 146 154  
 Gaussian kernel, 266 274  
 Gaussian Mixture PHD filter, 271  
 generalized linear models, 116  
 generative model, 113 157 160 161 275  
 Gestalt, 4  
 gesture interface, 233 235 236 238 247 253 257  
 Gibbs distribution, 75  
 Gibbs energy, 53  
 global context, 112 124  
 global functional structure, 4  
 global neural structure, 3  
 Graph Cuts, 51 53 59 61 63 65 69 71 73 75 77 81 83 85 87 89 91 93 95 97 99 101 103 105 107
- Hierarchical CRF, 122 123 127 129 132  
 homographic transformation, 205 212 214 218 219  
 human pose estimation, 51 77  
 human vision, 1-3 5 7 9 11 13 15 17 19 21 23 25
- image categorization, 173 174 191 192 200  
 image labeling, 124  
 image level prior, 192 194  
 image restoration, 51 54
- image segmentation, 51 52 58 59 62 67 71 73 75 78 79 88 93 96 103 112 123 174 281  
 image understanding, 109 201  
 intentionality of visual experiences, 4  
 Invariance, 6 28 178 180 197 200 244 282  
 Ising model, 54 113 117 118 128  
 isotropic dimension, 8
- K-means, 166 214 275 278  
 k-nearest neighbor, 256  
 Kalman filter, 271 275  
 Karhunen-Loëve, 139  
 Karhunen-Loëve eigenvalue, 135  
 Karhunen-Loëve expansion, 139
- LDA, 167 169 242 243 248 251 256  
 linear discriminant analysis, 242 256  
 linkage structure, 210 212 217 219 221  
 local context, 112 124 130  
 local features, 27 30 33 44  
 low level vision, 51 52 103
- Markov Chain Monte Carlo, 119  
 Markov random fields, 52 53 109 112 288  
 matching, 30 77 131 138 175 200 207 214 217 276 287 291 295 304 307 319 329 330  
 max-flow, 54 55 58 62 64 65 67 73 98 99 121  
 MAXCUT, 53  
 maximum a posteriori, 52-54 75 121 190 191  
 maximum likelihood, 119 123 128  
 Mean-shift, 275 278  
 Mixture of Gaussians, 275 278  
 motion capture, 237  
 MRF, 53 54 67 70 75 94 95 100 104 109 112 117 121 128 129 286 288 290 292 295 299 300 325 330
- multi-view object models, 205  
 multi-view photometric stereo, 330  
 multi-view stereo, 281 283 287 289 291 293 295 297 299 301 303 305 307 309 311 340 341  
 multi-view tracking, 263 269 272 274 277 278

- multiple similarity, 139, 142  
Mutual Information, 284  
Mutual Subspace Method, 135, 137–139, 141–149, 151, 153–155
- normalized cross correlation, 241, 284, 287  
nuisance, 27, 30, 206, 209, 228
- object categorization, 157, 160, 183, 206, 209  
object detection, 77, 82–84, 109, 111, 112, 115, 122, 125, 127, 129, 263  
object pose, 205–207  
object recognition, 157–161, 168, 200  
object reconstruction, 51, 88, 89, 209, 285  
object representation, 210, 211  
OCR, 135  
online classification, 240, 242  
optical flow, 41, 237, 238, 240, 241, 248
- PCA, 135, 139, 238  
photometric stereo, 313, 321, 323, 325, 327, 329, 333, 335, 337, 339, 341, 343, 345  
pictorial structure models, 210  
pose estimation, 51, 74, 75, 77, 78, 85, 86, 91, 93, 205, 207, 209, 211, 213, 215, 217, 219, 221, 225, 227, 229, 231  
Potts model, 69, 101, 113, 123, 130, 325  
Principal Component Analysis, 135, 139, 236  
principle of global selectivity, 3  
principle of local selectivity, 3  
principle of locality, 3  
pyramid match kernel, 174, 191, 192
- randomized decision forests, 173, 175  
RANSAC, 213, 214, 266  
recognition, 27, 29, 52, 77, 80, 109, 135, 139, 144, 148, 154, 157–161, 166, 169, 170, 187, 200, 205, 207, 209, 210, 212, 219, 226, 228, 235, 238, 257  
Reconstruction, 51, 87–89, 91, 209, 283–286, 292, 299, 306, 313–315, 317, 321, 323, 327, 329, 333, 335, 337, 343, 345  
region matching, 240–242  
Regularisation, 291, 293, 299, 301, 302, 320, 322, 324, 326, 327, 341
- reparameterization, 56, 60–63  
residual graph, 56, 60–66, 71, 96, 98–101  
rigid motion, 46  
ROC curves, 147
- saddle point approximation, 120, 123  
saliency detector, 213, 220  
scene classification, 157, 161  
scene recognition, 159, 161, 169, 187  
Segmentation, 51, 52, 54, 58, 59, 62, 67–71, 73–80, 82, 84, 89, 91–93, 96, 103, 109, 112, 113, 122, 123, 166, 173, 174, 181, 188, 189, 192, 194–196, 198, 201, 275, 276, 281, 283, 286, 292, 293, 307, 326  
semantic segmentation, 173, 174, 188, 192, 200, 201  
semantic texton forest, 173, 174, 179, 182, 187, 190, 191, 200, 201  
shading, 282, 283, 313, 321, 326, 327, 329, 333  
shape from shading, 9, 13, 14, 16, 18–20  
shape priors, 75, 77, 78, 87, 88, 92  
shape-from-photographs, 281  
Sherlock model, 5  
SIFT, 30, 31, 36, 39, 41, 42, 44, 161, 166, 174, 187, 213, 220, 266  
silhouette, 282, 293, 306, 329–334, 338, 340  
single instance 3D models, 207, 209  
single-view 2D models, 207  
Stickman model, 76, 79, 83  
structure of image space, 11  
Subspace Method, 135, 137–139, 141–143, 145, 147, 149, 151, 153, 155  
Sum of Square Differences, 284
- template matching, 135, 139, 236, 240, 241, 244, 248  
Tensor Canonical Correlation, 155  
Textons, 173, 174, 183, 191, 192, 194–197, 199, 200  
texture, 282, 288, 302, 305, 307, 328, 329, 332  
topological structure, 10  
touch-free interaction, 233  
tracking, 233, 240, 243–245, 247, 250–252, 254, 255, 257, 263, 267, 269, 271, 279  
transparency, 282  
Upper body model, 79, 83, 86

- view-synthesis, 208 220 222 224 226  
227  
vignetting, 13 15  
vision-based remote control, 233 235 237  
239 241 243 245 247 249 251  
253 255 257 259 261  
visual attention mechanism, 253
- visual field, 6 8  
visual recognition, 27 28 157 158 170  
206  
visual system, 2 5 12 15 16 21 23 24  
27 157 206  
visual words, 166 174  
vocabulary, 214 237