Studies in Computational Intelligence 411

Roberto Cipolla
Sebastiano Battiato
Giovanni Maria Farinella (Eds.)

Machine Learning for Computer Vision

Springer

Studies in Computational Intelligence 411

For further volumes:
http://www.springer.com/series/7092

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

Machine Learning for Computer Vision

Springer

Editors
Prof. Roberto Cipolla
Department of Engineering
University of Cambridge
Cambridge
UK

Dr. Giovanni Maria Farinella
Dipartimento di Matematica ed Informatica
Università di Catania
Catania
Italy

Prof. Sebastiano Battiato
Dipartimento di Matematica ed Informatica
Università di Catania
Catania
Italy

ISSN 1860-949X
e-ISSN 1860-9503
ISBN 978-3-642-28660-5
e-ISBN 978-3-642-28661-2
DOI 10.1007/978-3-642-28661-2
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012933083

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 recent editions of the school and covering some of the key topics of machine learning for computer vision. The chapters provide both an in-depth overview of challenging areas and key references to the existing literature. The book starts with two chapters devoted to introducing the reader to the exciting fields of Machine Learning and Computer Vision. In Chapter 1 the problem of producing truly intelligent machines is discussed and the "neuromorphic" approach is introduced. Chapter 2 introduces the notion of visual information and its relevant properties useful to accomplish visual inference tasks. Chapter 3 reviews algorithms to rapidly search images or videos in large collections, whereas Chapter 4 discusses the problem of object recognition and introduces generative models able to take into account transformations of geometry and reflectance. Chapter 5 describes a method to quickly and accurately predict 3D positions of body joints from a single depth image. Chapter 6 describes a fast vote-based approach for 3D shape recognition and registration. Chapter 7 introduces novel multi-classifier boosting algorithms for object detection, tracking and segmentation tasks, whereas in Chapter 8 the problem of tracking objects using multiple cameras is described together with related

algorithms. Finally, Chapter 9 complete the book by presenting a vision system for the challenging task of autonomous driving vehicles.

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

Sicily, January 2012

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.

List of Contributors

Ryan P. Adams
University of Toronto
10 King's College Road, Toronto, ON, M5S 3G4, CA
e-mail: radams@psi.toronto.edu

Andrew Blake
Microsoft Research,
7 J. J. Thomson Ave, CB3 0FB, Cambridge, UK
e-mail: ablake@microsoft.com

Alberto Broggi
VisLab.it - Artificial Vision and Intelligent Systems Lab
Dipartimento di Ingegneria dell'Informazione
University of Parma
Parco Area delle Scienze 181A, I-43100 Parma, IT
e-mail: broggi@vislab.it

Stefano Cattani
VisLab.it - Artificial Vision and Intelligent Systems Lab
Dipartimento di Ingegneria dell'Informazione
University of Parma
Parco Area delle Scienze 181A, I-43100 Parma, IT
e-mail: cattani@vislab.it

Andrea Cavallaro
School of Electronic Engineering and Computer Science
Queen Mary University of London
Mile End Road, London E1 4NS, UK
e-mail: andrea.cavallaro@elec.qmul.ac.uk

Jeroen C. Chua
University of Toronto
10 King's College Road, Toronto, ON, M5S 3G4, CA
e-mail: jeroen@psi.toronto.edu

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

Mat Cook
Microsoft Research,
7 J. J. Thomson Ave, CB3 0FB, Cambridge, UK

Rob Fergus
Deptartment of Computer Science
New York University
715 Broadway, New York, NY 10003, USA
e-mail: fergus@cs.nyu.edu

Mark Finocchio
Microsoft Research,
7 J. J. Thomson Ave, CB3 0FB, Cambridge, UK

Andrew Fitzgibbon
Microsoft Research,
7 J. J. Thomson Ave, CB3 0FB, Cambridge, UK
e-mail: awf@microsoft.com

Brendan J. Frey
University of Toronto
10 King's College Road, Toronto, ON, M5S 3G4, CA
e-mail: frey@psi.toronto.edu

Riccardo Gherardi
Computer Vision Group
Toshiba Research Cambridge Ltd
208 Cambridge Science Park, Cambridge CB4 0GZ, UK
e-mail: riccardo.gherardi@crl.toshiba.co.uk

Inmar E. Givoni
University of Toronto
10 King's College Road, Toronto, ON, M5S 3G4, CA
e-mail: inmar@psi.toronto.edu

Kristen Grauman
Deptartment of Computer Science
University of Texas at Austin
1616 Guadalupe, Suite 2.408 Austin, TX 78701, USA
e-mail: grauman@cs.utexas.edu

Alex Kipman
Microsoft Research,
7 J. J. Thomson Ave, CB3 0FB, Cambridge, UK

Tae-Kyun Kim
Department of Electrical and Electronic Engineering
Imperial College, London
South Kensington Campus, London SW7 2AZ, UK
e-mail: tk.kim@imperial.ac.uk

Joel Z. Leibo
McGovern Institute for Brain Research
Massachusetts Institute of Technology
43 Vassar Street, Cambridge, Massachusetts, USA
e-mail: jzleibo@mit.edu

Atsuto Maki
Computer Vision Group
Toshiba Research Cambridge Ltd
208 Cambridge Science Park, Cambridge CB4 0GZ, UK
e-mail: atsuto.maki@crl.toshiba.co.uk

Paolo Medici
VisLab.it - Artificial Vision and Intelligent Systems Lab
Dipartimento di Ingegneria dell'Informazione
University of Parma
Parco Area delle Scienze 181A, I-43100 Parma, IT
e-mail: medici@vislab.it

Richard Moore
Microsoft Research,
7 J. J. Thomson Ave, CB3 0FB, Cambridge, UK

Frank Perbet
Computer Vision Group
Toshiba Research Cambridge Ltd
208 Cambridge Science Park, Cambridge CB4 0GZ, UK
e-mail: frank.perbet@crl.toshiba.co.uk

Minh-Tri Pham
Computer Vision Group
Toshiba Research Cambridge Ltd
208 Cambridge Science Park, Cambridge CB4 0GZ, UK
e-mail: mtpham@crl.toshiba.co.uk

Tomaso Poggio
McGovern Institute for Brain Research
Massachusetts Institute of Technology
43 Vassar Street, Cambridge, Massachusetts, USA
e-mail: tp@csail.mit.edu

Toby Sharp
Microsoft Research,
7 J. J. Thomson Ave, CB3 0FB, Cambridge, UK

Jamie Shotton
Microsoft Research,
7 J. J. Thomson Ave, CB3 0FB, Cambridge, UK
e-mail: 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
e-mail: soatto@ucla.edu

Björn Stenger
Computer Vision Group
Toshiba Research Cambridge Ltd
208 Cambridge Science Park, Cambridge CB4 0GZ, UK
e-mail: bjorn@cantab.net

Murtaza Taj
School of Electronic Engineering and Computer Science
Queen Mary University of London
Mile End Road, London E1 4NS, UK
e-mail: murtaza.taj@elec.qmul.ac.uk

Cheston Tan
McGovern Institute for Brain Research
Massachusetts Institute of Technology
43 Vassar Street, Cambridge, Massachusetts, USA
e-mail: cheston@mit.edu

Oliver J. Woodford
Computer Vision Group
Toshiba Research Cambridge Ltd
208 Cambridge Science Park, Cambridge CB4 0GZ, UK
e-mail: oliver.woodford@crl.toshiba.co.uk

Paolo Zani
VisLab.it - Artificial Vision and Intelligent Systems Lab
Dipartimento di Ingegneria dell'Informazione
University of Parma
Parco Area delle Scienze 181A, I-43100 Parma, IT
e-mail: zani@vislab.it

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 a Fellow at the Royal Academy of Engineering (since 2010); also a Professor of Computer Vision at the Royal Academy of Arts, London (since 2004) and 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 received his degree in computer science (summa cum laude) in 1995 from University of Catania and his Ph.D. in computer science and applied mathematics in 1999 from University of Naples. From 1999 to 2003 he was the leader of the "Imaging" team at STMicroelectronics in Catania. He joined the Department of Mathematics and Computer Science at the University of Catania as assistant professor in 2004 and became associate professor in the same department in

2011. His research interests include image enhancement and processing, image coding, camera imaging technology and multimedia forensics. He has edited 4 books and co-authored more than 120 papers in international journals, conference proceedings and book chapters. He is a co-inventor of about 15 international patents, reviewer for several international journals, and he has been regularly a member of numerous international conference committees. Prof. Battiato has participated in many international and national research projects. Chair of several international events (ECCV2012, VISAPP2012, ICIAP 2011, ACM MiFor 2010-2011, SPIE EI Digital Photography 2011-2012, etc.). He is an associate editor of the IEEE Transactions on Circuits and System for Video Technology and of the SPIE Journal of Electronic Imaging. Guest editor of the following special issues: "Emerging Methods for Color Image and Video Quality Enhancement" published on EURASIP Journal on Image and Video Processing (2010) and "Multimedia in Forensics, Security and Intelligence" published on IEEE Multimedia Magazine (2012). He is director (and co-founder) of the International Computer Vision Summer School (ICVSS), Sicily, Italy. He is a senior member of the IEEE.

Giovanni Maria Farinella

Giovanni Maria Farinella obtained the Master 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 Image Processing Laboratory (IPLAB) at the Department of Mathematics and Computer Science, University of Catania in 2008 as a Contract Researcher. He became Associate Member of the Computer Vision and Robotics Research Group at University of Cambridge in 2006. He is Contract Professor of Computer Vision at the Academy of Arts of Catania (since 2004) and Adjunct Professor of Computer Science at the School of Medicine of the University of Catania (since 2011). His research interests lie in the fields of Computer Vision, Pattern Recognition and Machine Learning. He has edited two volumes and co-authored more than 50 papers in international journals, conference proceedings and book chapters. He is a co-inventor of 2 international patents. He also serves as a reviewer and on the programme committee for major international journals and international conferences. Dr. Farinella founded (in 2006) and currently directs the International Computer Vision Summer School (www.dmi.unict.it/icvss).

Contents

Throwing Down the Visual Intelligence Gauntlet

Cheston Tan, Joel Z. Leibo, and Tomaso Poggio

Abstract. In recent years, scientific and technological advances have produced artificial systems that have matched or surpassed human capabilities in narrow domains such as face detection and optical character recognition. However, the problem of producing truly intelligent machines still remains far from being solved. In this chapter, we first describe some of these recent advances, and then review one approach to moving beyond these limited successes – the *neuromorphic* approach of studying and reverse-engineering the networks of neurons in the human brain (specifically, the visual system). Finally, we discuss several possible future directions in the quest for visual intelligence.

1 Artificial Intelligence: Are We There Yet?

Every few years (and sometimes more often), the world becomes abuzz with excitement over some new technology that, finally, after all these years, promises to fulfill the dream of Artificial Intelligence (AI), first proposed back in 1956 at the dawn of the Age of Computing. The latest of these technologies is *Watson*, a natural language question-answering computer system that, in February 2011, defeated two of the best human contestants ever in the quiz show *Jeopardy!*

In 2007, the buzz-worthy news was that six teams successfully completed the DARPA Grand Challenge [3], a simulated 60-mile urban course designed to test the capabilities of driverless vehicle systems. The overall winner completed the course just over 4 hours, averaging approximately 14 mph. Just two years prior, in 2005, five teams completed the 132-mile off-road desert course, a stunning reversal of the 2004 results, in which the best vehicle only managed to complete an embarrassing 7.36 miles of the 150-mile course [3].

Cheston Tan · Joel Z. Leibo · Tomaso Poggio
McGovern Institute for Brain Research at Massachusetts Institute of Technology, Cambridge, MA
e-mail: {cheston,jzleibo,tp}@mit.edu

R. Cipolla et al. (Eds.): Machine Learning for Computer Vision, SCI 411, pp. 1–15.
springerlink.com

Apart from these and other headline-grabbing milestones like chess-playing computer Deep Blue defeating the reigning world chess champion in 1997, there have also been a host of less-publicized technological advances that have matched (and sometimes even surpassed) human abilities. These include face-detection technology in digital cameras [4], a pedestrian-detection feature in the latest luxury car models [1, 6], and optical character recognition (OCR) technology used by postal services [9] around the world, just to name a few.

Meanwhile, in the realm of computer vision, performance of algorithms on common datasets such as Caltech-101 [2] and PASCAL [8] have improved steadily over the years. While typical human performance on these datasets has not been quantified, it is not unimaginable that computers may reach this performance benchmark in a decade or less.

Given this plethora of advances and achievements that would be utterly jaw-dropping to the early pioneers of computing and AI, should we thus conclude that victory in achieving AI is close at hand? Our answer is a clear ***no***. None of these systems or computers can really be described as intelligent in the way that one would describe a person. Each of these systems performs well in a narrow domain such as categorizing objects or playing chess, but two key characteristics of human intelligence are broadness and flexibility. A typical person is capable of learning not only chess, but hundreds of other games. With sufficient training, a typical person would also be able to become good at any of these.

One might argue that if various artificial systems achieve human-level performance in a sufficient number of these narrow domains, then putting these components together in a single system would result in some semblance of human intelligence. We think otherwise – such a system may fulfill the criterion of broadness, but it would lack flexibility.

We do not mean to downplay the amazing advances in computing that have occurred and are occurring. Computers have changed our lives for the better to an extent that almost no other technology has, and the advances mentioned above will no doubt further these changes in exciting and dramatic ways. The ongoing pursuit of tangible engineering solutions to pressing challenges is an important and worthwhile research agenda. However, the goal of replicating broad and flexible human-like intelligence will not be achieved just by stringing together the solutions to specialized problems.

1.1 A Compass for the Uncharted Journey towards Intelligence

One approach (arguably the default one) to move beyond the limitations of tackling narrow domains in isolation, is to study the brain – a computational system which we already know exhibits intelligence. However, much of the computation in the brain is still poorly understood. Understanding how the individual functioning of billions of brain cells leads collectively to human intelligence and cognition remains a major challenge.

Nonetheless, a good bet – and the one we make – is that in order to rise to the challenge, we need to understand how the *visual cortex* works, and then reproduce it in computers. Vision is one of the most studied aspects of human cognition, and over one-third of the cerebral cortex (the seat of cognition) is dedicated to visual processing. Replicating intelligence will ultimately require more than just understanding visual cognition, but it is likely to be the best place to start.

This *neuromorphic* approach does not imply the slavish, neuron-by-neuron reproduction of the human visual system. Just like how the marvels of modern flight have been enabled by the scientific understanding of the principles of aerodynamics, we believe that the key to unlocking intelligence is to investigate its principles by studying systems that truly exhibit intelligence (and to validate our understanding of these principles by building testable computational models).

In the rest of this chapter, we first briefly review some of the computational principles underlying visual intelligence in the cortex, and then proceed to sketch the first steps towards replicating these principles in computational models. Finally, we discuss several suggestions for moving research forward in the direction of true intelligence.

2 The Neuromorphic Approach to Visual Intelligence

By now, there are probably several hundred models of the visual cortex. Most deal with specific visual phenomena (such as visual illusions) or specific parts of the visual cortex. Many of these have yielded useful contributions to neuroscience. However, if the goal is to use these models to guide the engineering of a new generation of systems that approach human visual intelligence, then more comprehensive models addressing a wide range of phenomena and visual areas are needed. Thus, in this section, we review a computational model of visual cortex that fulfills precisely these criteria. This model (or more precisely, this class of models) provides a preliminary but illustrative sketch of how computations performed by the visual cortex are closely linked to the principles underlying visual intelligence (at least for two principles, among the few that are currently known). First, however, we briefly review relevant background knowledge regarding the visual cortex.

2.1 Anatomy and Physiology of the Primary Visual Cortex (V1)

Neural recordings from the *primary visual cortex* (also known as *V1*) of cats in the early 1960s by Nobel laureates[1] David Hubel and Torsten Wiesel yielded the observation of so-called *simple cells* responding to edges of a particular orientation. Hubel and Wiesel also described another class of cells with more complicated responses which came to be called *complex cells*. In the same publication [15], they

[1] One half of the 1981 Nobel Prize in Physiology or Medicine was jointly awarded to Hubel and Wiesel. The other half was awarded to Roger Sperry.

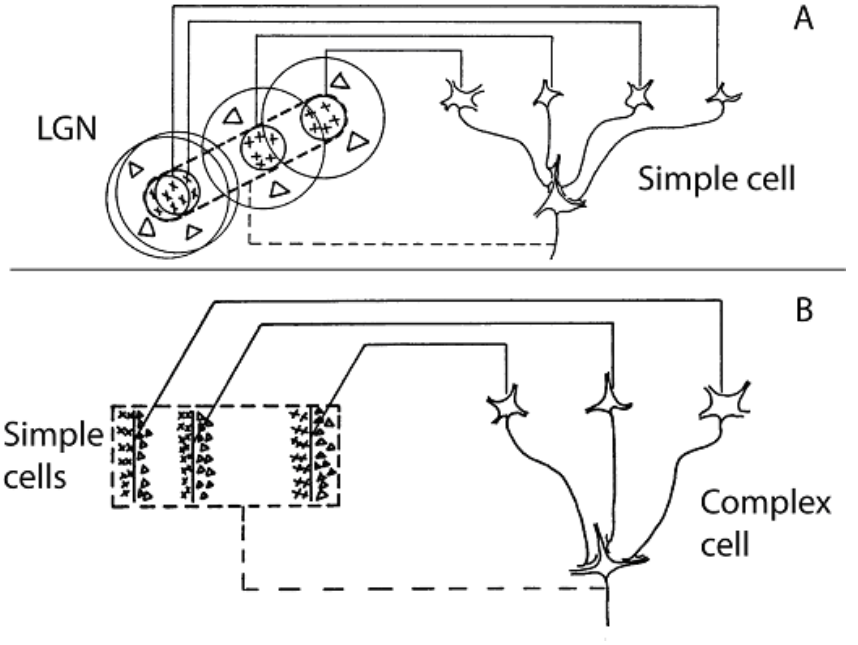


Fig. 1 Construction of V1 simple and complex cell receptive fields via convergent inputs. (Adapted from Hubel and Wiesel 1962)

hypothesized that (at least some of) the complex cells may be receiving their inputs from the simple cells.

The simple cells' receptive fields[2] contain oriented "on" regions in which presenting an appropriately-oriented edge stimulus excited the cells, and "off" regions for which presenting a stimulus suppresses neural activity. These classical Gabor-like receptive fields can be understood by noting that they are easily built from a convergence of inputs from cells in the *lateral geniculate nucleus* (LGN), a brain structure from which V1 receives strong inputs. The simple cells respond only when receiving simultaneous inputs from several LGN cells with receptive fields arranged along a line of the appropriate orientation. Fig. 1a is a reproduction of Hubel and Wiesel's original drawing from their 1962 publication illustrating the appropriate convergence of LGN inputs.

In contrast to simple cells, Hubel and Wiesel's complex cells respond to edges with particular orientations, but notably have no "off" regions where stimulus presentation reduces responses. Most complex cells also have larger receptive fields than simple cells, i.e., an edge of the appropriate orientation will stimulate the cell when presented anywhere over a larger region of space. Hubel and Wiesel noted that

[2] A cell's *receptive field* is a region of visual space in which the presence of a stimulus will affect the activity of that cell.

the complex cell fields could be explained by a convergence of inputs from simple cells. Fig. 1b reproduces their scheme.

Following Hubel and Wiesel, we say that the simple cells are tuned to a particular preferred feature. This tuning is accomplished by weighting the LGN inputs in such a way that a simple cell fires when the inputs arranged to build the preferred feature are co-activated. In contrast, the complex cells' inputs are weighted such that the activation of any of their inputs can drive the cell by itself. So the complex cells are said to pool the responses of several simple cells. As information about the stimulus passes from LGN to V1, its representation increases in selectivity; patterns without edges (such as sufficiently small circular dots of light) are no longer represented. Then, as information passes from simple cells to complex cells, the representation gains tolerance to the spatial position of the stimulus. Complex cells downstream from simple cells that respond only when their preferred feature appears in a small window of space, now represent stimuli presented over a larger region.

2.2 *Hubel-Wiesel Models: Successive Tuning and Pooling*

Beyond V1, visual cortex is broadly organized into two parallel processing streams, a *ventral* stream mostly involved in analysis of shape information, and a *dorsal* stream mostly involved in analysis of motion and location [23]. Both streams are organized hierarchically, with receptive field sizes and preferred feature complexity increasing along the way from their starting point in V1 to subsequent areas (see Fig. 2). The present chapter focuses on the ventral stream, but see [18] for related models of the dorsal stream.

At the top of the ventral hierarchy, the cells in *inferotemporal* (IT) cortex respond selectively to highly complex stimuli, and invariantly over several degrees of visual angle. Hierarchical models inspired by the work of Hubel and Wiesel [10, 13, 22, 25, 28, 30, 32, 37, 39], henceforth termed ***H-W models*** ([28, 30, 32] were previously known as ***HMAX***), seek to achieve similar selectivity and invariance properties by subjecting visual inputs to successive – and often alternating – tuning and pooling operations, just like how V1 simple cells are tuned to specific patterns of LGN inputs and complex cells pool the responses of simple cells with similar orientation selectivity. A major algorithmic claim made by these H-W models is that the repeated application of AND-like tuning is the source of the selectivity in IT cortex. Likewise, repeated application of OR-like pooling produces invariant responses.

H-W models nicely illustrate the close correspondence between visual intelligence and the visual cortex. Through careful scientific study, the idealized tuning and pooling operations in H-W models have been distilled from the jumble of neural activity in the visual cortex. These operations are, not surprisingly, associated with the principles of selectivity and invariance that underlie object recognition – an important component of visual intelligence.

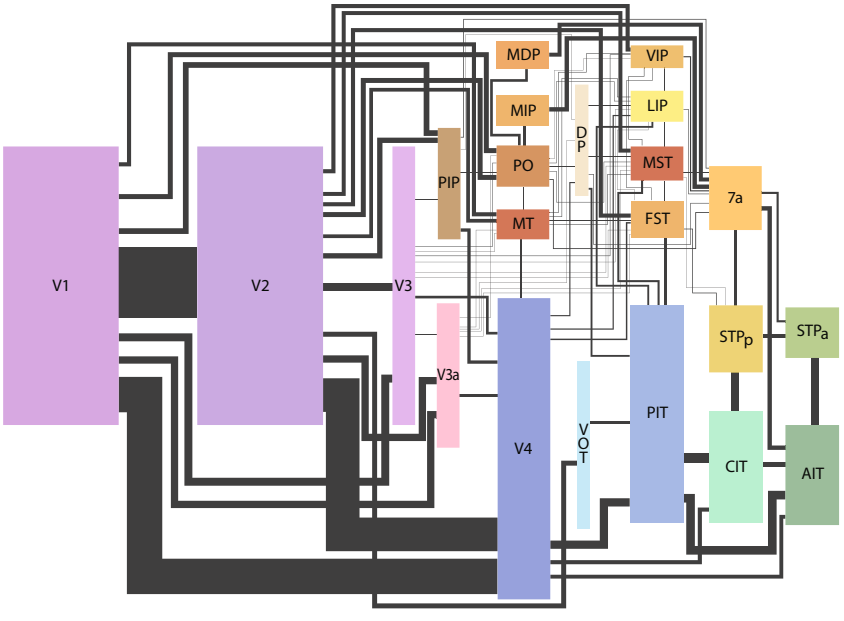


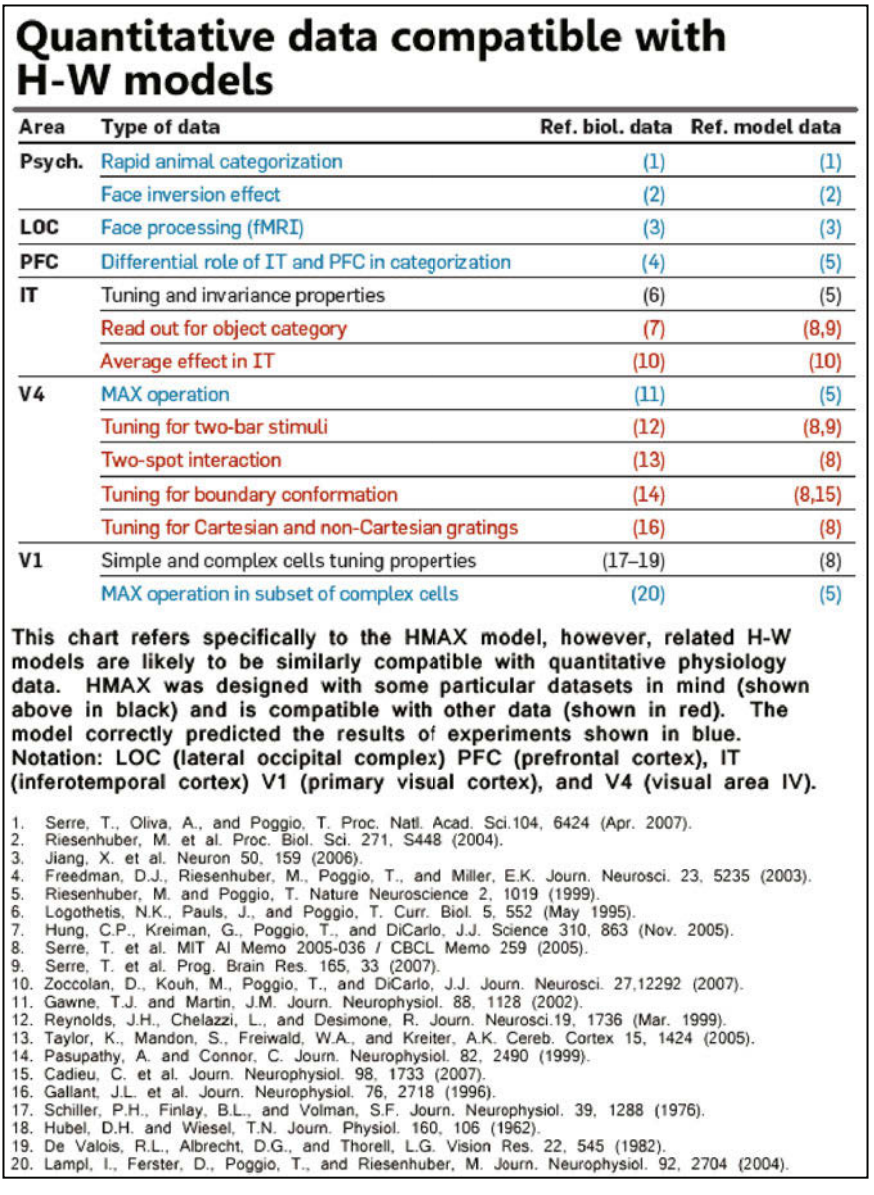
Fig. 2 Areas of the visual cortex. Each rectangle represents a visual area; the size is proportional to cortical surface area. The lines connecting the areas have a thickness proportional to the estimated number of fibers in the connection. Ventral stream areas are in the bottom half; dorsal stream areas are in the upper half. (Reprinted from [38])

2.3 Consistency with Experimental Results at Multiple Levels

H-W models were constructed based on the dual principles of selectivity and invariance, derived initially from the study of V1 but also subsequently found in other visual areas. The key test of any model, however, is the consistency of its predictions with phenomena beyond those used in its construction.

In that regard, recent H-W models are consistent with experimental findings at multiple levels of analysis, from the computational to the biophysical level. Being consistent across all these levels is a high bar and an important one. It is relatively easy to develop models that just explain one phenomenon or one illusion or one visual area, but they remain just that: a model of that one thing. Such narrow models are not useful for guiding future research on the general problems of vision and intelligence.

Predictions of H-W models (in particular, the HMAX model [28, 30, 32]), have generally fared well when checked against experimental results. In this section, we briefly review evidence from electrophysiology and psychophysics in relation to this class of models. A more comprehensive list of comparisons between experimental and model data can be found in Fig. 3 (also see [29, 30] for more details).



Quantitative data compatible with H-W models

Area	Type of data	Ref. biol. data	Ref. model data
Psych.	Rapid animal categorization	(1)	(1)
	Face inversion effect	(2)	(2)
LOC	Face processing (fMRI)	(3)	(3)
PFC	Differential role of IT and PFC in categorization	(4)	(5)
IT	Tuning and invariance properties	(6)	(5)
	Read out for object category	(7)	(8,9)
	Average effect in IT	(10)	(10)
V4	MAX operation	(11)	(5)
	Tuning for two-bar stimuli	(12)	(8,9)
	Two-spot interaction	(13)	(8)
	Tuning for boundary conformation	(14)	(8,15)
	Tuning for Cartesian and non-Cartesian gratings	(16)	(8)
V1	Simple and complex cells tuning properties	(17–19)	(8)
	MAX operation in subset of complex cells	(20)	(5)

This chart refers specifically to the HMAX model, however, related H-W models are likely to be similarly compatible with quantitative physiology data. HMAX was designed with some particular datasets in mind (shown above in black) and is compatible with other data (shown in red). The model correctly predicted the results of experiments shown in blue. Notation: LOC (lateral occipital complex) PFC (prefrontal cortex), IT (inferotemporal cortex) V1 (primary visual cortex), and V4 (visual area IV).

1. Serre, T., Oliva, A., and Poggio, T. Proc. Natl. Acad. Sci.104, 6424 (Apr. 2007).
2. Riesenhuber, M. et al. Proc. Biol. Sci. 271, S448 (2004).
3. Jiang, X. et al. Neuron 50, 159 (2006).
4. Freedman, D.J., Riesenhuber, M., Poggio, T., and Miller, E.K. Journ. Neurosci. 23, 5235 (2003).
5. Riesenhuber, M. and Poggio, T. Nature Neuroscience 2, 1019 (1999).
6. Logothetis, N.K., Pauls, J., and Poggio, T. Curr. Biol. 5, 552 (May 1995).
7. Hung, C.P., Kreiman, G., Poggio, T., and DiCarlo, J.J. Science 310, 863 (Nov. 2005).
8. Serre, T. et al. MIT AI Memo 2005-036 / CBCL Memo 259 (2005).
9. Serre, T. et al. Prog. Brain Res. 165, 33 (2007).
10. Zoccolan, D., Kouh, M., Poggio, T., and DiCarlo, J.J. Journ. Neurosci. 27,12292 (2007).
11. Gawne, T.J. and Martin, J.M. Journ. Neurophysiol. 88, 1128 (2002).
12. Reynolds, J.H., Chelazzi, L., and Desimone, R. Journ. Neurosci.19, 1736 (Mar. 1999).
13. Taylor, K., Mandon, S., Freiwald, W.A., and Kreiter, A.K. Cereb. Cortex 15, 1424 (2005).
14. Pasupathy, A. and Connor, C. Journ. Neurophysiol. 82, 2490 (1999).
15. Cadieu, C. et al. Journ. Neurophysiol. 98, 1733 (2007).
16. Gallant, J.L. et al. Journ. Neurophysiol. 76, 2718 (1996).
17. Schiller, P.H., Finlay, B.L., and Volman, S.F. Journ. Neurophysiol. 39, 1288 (1976).
18. Hubel, D.H. and Wiesel, T.N. Journ. Physiol. 160, 106 (1962).
19. De Valois, R.L., Albrecht, D.G., and Thorell, L.G. Vision Res. 22, 545 (1982).
20. Lampl, I., Ferster, D., Poggio, T., and Riesenhuber, M. Journ. Neurophysiol. 92, 2704 (2004).

Fig. 3 Summary of comparisons between data from biological experiments and from the HMAX model. (Adapted from [31])

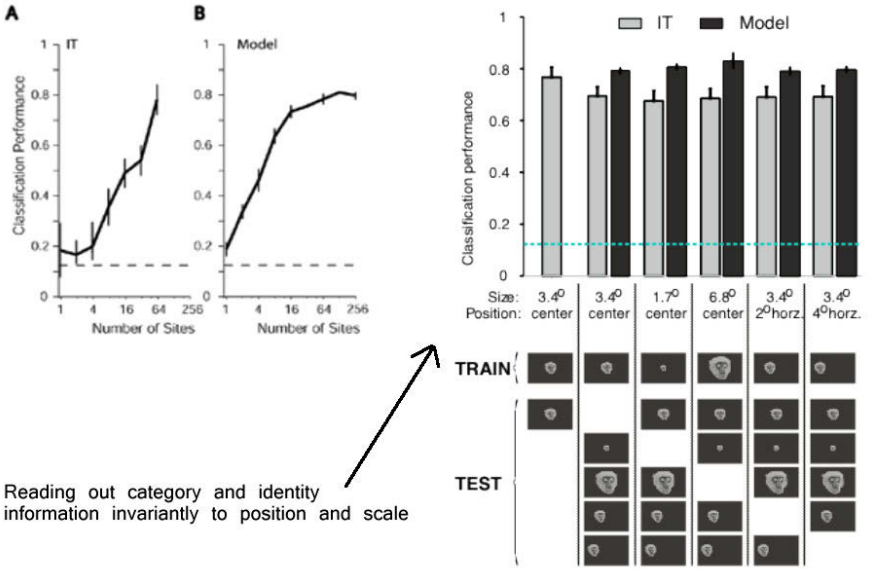


Fig. 4 Comparison of position- and scale-invariant decoding of stimulus category from populations of IT and model units. Both generalize fairly well when tested on novel positions and scales (TEST), after training on objects of a specific position and scale (TRAIN). See [16, 29] for more details. (Adapted from [16, 29])

Beginning in the primary visual cortex, an electrophysiology study in cats uncovered evidence that the brain employs an OR-like pooling operation, predicted by Riesenhuber and Poggio [28] as the mechanism by which complex cell receptive fields are built from simple cells [20]. Further evidence for this operation was independently found in area V4 (a visual area that receives inputs from V1) in rhesus monkeys [14], confirming that this key operation exists in multiple visual areas.

In addition, single-unit electrophysiology experiments in area V2 revealed that neurons in this area are sensitive to combinations of orientations [11], consistent with the AND-like tuning operation in the model [28]. Furthermore, a quantitative fit was established between H-W model units and the firing rates of V4 neurons evoked by a library of stimuli. This model was fit with data from a subset of cells, and could generalize to predict the responses of other cells to novel stimuli [12].

H-W models have been particularly influential in the study of IT cortex, the visual area situated at the top of the hierarchy of shape-processing areas. It is possible to decode object category information invariantly to translation and scaling from the responses of a population of IT cells [16]. The top-most layer of an H-W model also supports similar decoding (see Fig. 4). It is this observation that populations of IT cells and H-W model units both provide useful representations for decoding stimulus information which motivates much of the interest in IT cortex from the computer vision and neuroscience communities.

Importantly, H-W models demonstrate human performance levels on certain psychophysical tasks. Human observers can categorize scenes containing a particular prominent object, such as an animal or a vehicle, after only 20 ms of exposure. Old EEG experiments in humans, but also especially new data obtained with a "read-out" information decoding technique, show category information in the neural population of IT of the macaque at 80ms after a stimulus is seen. These experimental results establish a lower bound on the latency of visual categorization decisions made by the human visual system, and suggest that categorical decision-making can be implemented by a feedforward information processing mechanism like an H-W model [19, 21, 33, 34, 36]. Serre et al. showed that a specifc H-W model does indeed reach human-level performance on this task [30]. Many of the individual images on which the model failed the task were also the most difficult for humans to discriminate – suggesting a deep correspondence between the model's mechanisms and those implemented by human visual cortex (Fig. 5).

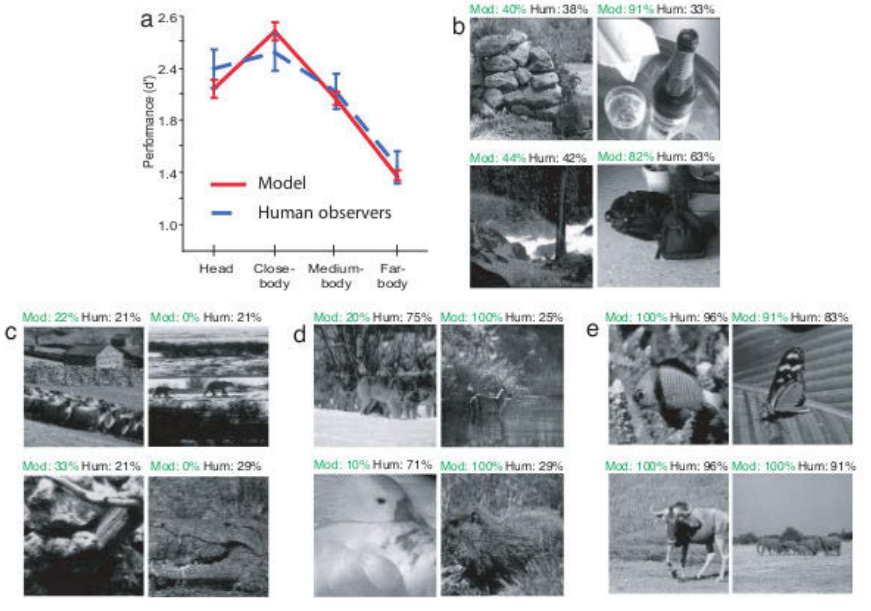


Fig. 5 Comparison between the model and humans on the task of categorizing rapidly-presented scenes as either containing an animal or not. (a) The model and humans exhibit a similar pattern of performance. (b–e) Examples of classifications by the model and human observers. The percentages above each thumbnail correspond to the number of times the image was classified as containing an animal by the model or by humans. Common false alarms (b) and misses (c) for the model and human observers. (d, e) Examples of animal images for which the agreement between the model and human observers is poor (d) and good (e). Overall correlations between human and model classification are 0.71, 0.84, 0.71 and 0.60 for the "Head", "Close-body", "Medium-body" and "Far-body" images respectively. (Reprinted from [30]. Copyright 2007, National Academy of Sciences, USA)

2.4 From Neuroscience Models to Engineering Applications

While H-W models clearly have a long way to go before they begin to approach human levels of broad visual intelligence, they have nonetheless shown success beyond the realm of neuroscience by proving useful for computer vision.

Recently, one H-W model based on the motion-processing dorsal stream of visual cortex has been found to match human performance on the task of recognizing and annotating mouse behaviors in video clips [17]. Many biological experiments dealing with genetically-modified strains of mice require laborious human annotation of many hours of video in order to quantitatively analyze effects of the various genetic modifications performed in the quest for understanding diseases such as autism and Parkinson's disease. The neuromorphic model was developed into a trainable, general-purpose system capable of automatically analyzing complex mouse behaviors, performing on par with humans and outperforming a current commercial, non-neuromorphic system [5].

Furthermore, the adherence to neuromorphism has not significantly disadvantaged H-W models in terms of performance on standard computer vision datasets. At various points in time, these models have matched or surpassed state-of-the-art systems on datasets such as CalTech101 [24, 32] and Labeled-Faces-in-the-Wild (LFW) [27, 26]. These results reinforce the belief that ultimately, the quest to understand the key properties of biological intelligence will be essential in producing truly intelligent artificial systems.

3 What's Next in the Quest for Visual Intelligence?

Thus far, we have argued for the neuromorphic approach to tackling the challenge of achieving human-level visual intelligence, and then reviewed the successes of this approach up to this point. In this final section, we briefly describe some suggestions as to how computational neuroscience and computer vision should proceed from this point onwards.

3.1 Going beyond "What Is Where"

Much of computer vision research today is geared towards the "what" and "where" problems. Specifically, "what" problems include the detection and categorization of objects and actions, while "where" problems include localization, segmentation and tracking. However, as alluded to earlier, although determining "what is where" in an image is an important part of vision, visual intelligence is much more than that.

Take, for instance, the task of connecting an Ethernet cable to a laptop computer. Visually, this simple task consists of several sub-tasks, including finding potential locations on the surface of the laptop where the cable's connector might fit, as well as determining which orientation the connector should be held at. Certainly, these

sub-tasks include traditional "what" and "where" problems (e.g., segmentation of the laptop; detecting and localizing a region that matches the connector shape), but the greater challenge is determining precisely what these sub-tasks should be.

3.2 From Perception to Abstraction

Another example of how visual intelligence goes beyond "what is where" involves abstract visual concepts. Take, for example, the concepts of "peaceful" or "crowded"; images can be classified (albeit somewhat subjectively) as being "peaceful" or not, and being "crowded" or not[3]. Yet, these abstract notions are not so much about specific objects and their locations, but rather about properties of the image as a whole.

Even specific objects themselves have abstract properties. Every object has a physical size in terms of pixels, but the property of *conceptual size* (i.e., "largeness") depends on several factors, including *inferred size* (by determining the corresponding real-world scale of the depicted scene, if applicable), as well as comparisons to typical sizes of other instances from the same object category.

People are also good at identifying somewhat abstract actions from single static images. Shimon Ullman gives the example of identifying *drinking* (see Fig. 6). Current systems can probably recognize that these images contain liquids, cups, glasses, bottles, hands, mouths, etc. However, such systems do not encode the higher level knowledge that one form of drinking consists of using a cup to bring liquid towards a mouth, or that alternative forms of drinking are also valid.

3.3 A Loose Hierarchy of Visual Tasks

The "what" and "where" problems can be considered to be predominantly perceptual. It is clear that visual intelligence spans a variety of problems, ranging from these perceptual ones, to the more abstract or cognitive ones described earlier. It may be useful to conceptualize or organize the challenge of visual intelligence into a loose hierarchy of visual tasks, with the "easy" perceptual tasks at the bottom, and the most abstract tasks at the top[4]. If the Turing Test [35] is a yardstick for general human intelligence, then this collection of visual tasks can be considered to be a measure of visual intelligence – a ***Visual Turing Test***, so to speak.

Such a big-picture view of things may be important for guiding research, particularly if the goal is to create a single system that can accomplish all the tasks. For instance, focusing on a single task such as object categorization may lead to certain algorithms or approaches being considered state-of-the-art. However, such

[3] Determining the images for which these classification tasks do not make sense in the first place is also part of visual intelligence.

[4] The idea being that, roughly speaking, only the more "intelligent" systems should be able to perform tasks near the top of the hierarchy.

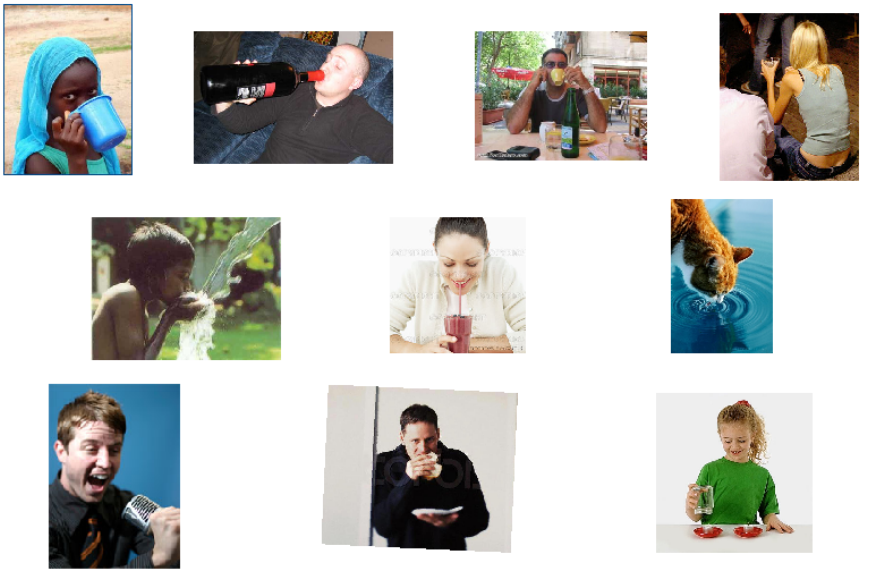


Fig. 6 All of these images contain *drinking*. Credit: Shimon Ullman

algorithms, by virtue of being extremely good at one thing, could inadvertently turn out to be poorly-suited for other tasks – falling into a local minimum, in some sense[5]. By laying out the bigger picture of tasks to be performed, research efforts are more likely to be directed towards approaches that are more general.

3.4 It's Time to Try again – The MIT Intelligence Initiative

Finally, we conclude this chapter by describing efforts in our lab and others at MIT to once again attempt to tackle the problem of intelligence, visual and otherwise – the MIT Intelligence Initiative [7].

The problem of intelligence – the nature of it, how the brain generates it and how it could be replicated in machines – is arguably one of the deepest and most important problems in science today. Philosophers have studied intelligence for centuries, but it is only in the last several decades that key developments in a broad range of science and engineering fields have opened up a thriving "intelligence research" enterprise, making questions such as these approachable: How does the mind process sensory information to produce intelligent behavior – and how can we design intelligent computer algorithms that behave similarly? What is the structure and form of human knowledge – how is it stored, represented and organized?

[5] These algorithms are of course still very valuable for solving specific problems.

Many of us at MIT believe that the time has come for a new, fresh attack on these problems. The launching off point will be a new integration of the fields of cognitive science, which studies the mind, neuroscience, which studies the brain, and computer science and artificial intelligence, which develop intelligent hardware and software. These fields grew up together in the 1950s, but drifted apart as each became more specialized. In the 21st century, they are re-converging as a result of new advances that allow studies of the brain and mind to inform the design of intelligent artifacts and vice versa. Hence, for the original, daring goals of understanding and replicating intelligence, perhaps it is time to try again.

Acknowledgements. This report describes research done at the Center for Biological & Computational Learning, which is in the McGovern Institute for Brain Research at MIT, as well as in the Dept. of Brain & Cognitive Sciences, and which is affiliated with the Computer Sciences & Artificial Intelligence Laboratory (CSAIL). This research was sponsored by grants from DARPA (IPTO and DSO), National Science Foundation (NSF-0640097, NSF-0827427), AFSOR-THRL (FA8650-05-C-7262). Additional support was provided by: Adobe, Honda Research Institute USA, King Abdullah University of Science and Technology grant to B. DeVore, NEC, Sony and especially by the Eugene McDermott Foundation.

References

1. A pedestrian detection system that stops a car automatically, http://articles.economictimes.indiatimes.com/2011-02-27/news/28638493_1_detection-system-volvo-collision-warning-system
2. Caltech 101, http://www.vision.caltech.edu/Image_Datasets/Caltech101/
3. DARPA Grand Challenge, http://en.wikipedia.org/wiki/DARPA_Grand_Challenge
4. Digital Camera Face Recognition: How It Works, http://www.popularmechanics.com/technology/how-to/4218937
5. HomeCageScan 2.0, http://www.cleversysinc.com/products/software/homecagescan
6. Night View Assist: How night becomes day., http://www.daimler.com/dccom/0-5-1210218-1-1210320-1-0-0-1210228-0-0-135-7165-0-0-0-0-0-0-0.html
7. The MIT Intelligence Initiative, http://isquared.mit.edu/
8. The PASCAL Visual Object Classes Homepage, http://pascallin.ecs.soton.ac.uk/challenges/VOC/
9. USPS Awards Parascript Contract for OCR to Support Automated Parcel Bundle Sorting at USPS Facilities throughout the United States, http://money.msn.com/business-news/article.aspx?feed=PR&Date=20110601&ID=13713512/
10. Amit, Y., Mascaro, M.: An integrated network for invariant visual detection and recognition. Vision Research 43(19), 2073–2088 (2003), http://dx.doi.org/10.1016/S0042-6989(03)00306-7, doi:10.1016/S0042-6989(03)00306-7
11. Anzai, A., Peng, X., Essen, D.V.: Neurons in monkey visual area V2 encode combinations of orientations. Nature Neuroscience 10(10), 1313–1321 (2007), http://www.nature.com/neuro/journal/vaop/ncurrent/full/nn1975.html

12. Cadieu, C., Kouh, M., Pasupathy, A., Connor, C., Riesenhuber, M., Poggio, T.: A model of V4 shape selectivity and invariance. Journal of Neurophysiology 98(3), 1733 (2007), http://jn.physiology.org/content/98/3/1733.short
13. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36(4), 193–202 (1980), http://www.springerlink.com/content/r6g5w3tt54528137, doi:10.1007/BF00344251
14. Gawne, T.J., Martin, J.M.: Responses of primate visual cortical V4 neurons to simultaneously presented stimuli. Journal of Neurophysiology 88(3), 1128 (2002), http://jn.physiology.org/content/88/3/1128.short
15. Hubel, D., Wiesel, T.: Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of Physiology 160(1), 106 (1962), http://jp.physoc.org/content/160/1/106.full.pdf
16. Hung, C.P., Kreiman, G., Poggio, T., DiCarlo, J.J.: Fast Readout of Object Identity from Macaque Inferior Temporal Cortex. Science 310(5749), 863–866 (2005), http://www.sciencemag.org/cgi/content/abstract/310/5749/863, doi:10.1126/science.1117593
17. Jhuang, H., Garrote, E., Yu, X., Khilnani, V., Poggio, T., Steele, A., Serre, T.: Automated home-cage behavioural phenotyping of mice. Nature Communications 1(6), 1–9 (2010), http://www.nature.com/ncomms/journal/v1/n6/abs/ncomms1064.html
18. Jhuang, H., Serre, T., Wolf, L., Poggio, T.: A biologically inspired system for action recognition. In: International Conference on Computer Vision (ICCV), vol. 11, pp. 1–8 (2007), http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4408988
19. Keysers, C., Xiao, D., Földiák, P., Perrett, D.: The speed of sight. Journal of Cognitive Neuroscience 13(1), 90–101 (2001), http://www.mitpressjournals.org/doi/abs/10.1162/089892901564199
20. Lampl, I., Ferster, D.: Intracellular measurements of spatial integration and the MAX operation in complex cells of the cat primary visual cortex. Journal of Neurophysiology 92(5), 2704 (2004), http://jn.physiology.org/content/92/5/2704.short
21. Li, F., VanRullen, R., Koch, C., Perona, P.: Rapid natural scene categorization in the near absence of attention. Proceedings of the National Academy of Sciences of the United States of America 99(14), 9596 (2002), http://www.pnas.org/content/99/14/9596.short
22. Mel, B.W.: SEEMORE: Combining Color, Shape, and Texture Histogramming in a Neurally Inspired Approach to Visual Object Recognition. Neural Computation 9(4), 777–804 (1997), http://dx.doi.org/10.1162/neco.1997.9.4.777, http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.4.777, doi:10.1162/neco.1997.9.4.777
23. Mishkin, M., Ungerleider, L.G., Macko, K.A.: Object vision and spatial vision: Two cortical pathways. Trends in Neurosciences 6, 414–417 (1983)
24. Mutch, J., Lowe, D.: Multiclass Object Recognition with Sparse, Localized Features. In: 2006 IEEE Conference on Computer Vision and Pattern Recognition, pp. 11–18. IEEE (2006), http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1640736, doi:10.1109/CVPR.2006.200
25. Perrett, D., Oram, M.: Neurophysiology of shape processing. Image and Vision Computing 11(6), 317–333 (1993), http://linkinghub.elsevier.com/retrieve/pii/0262885693900115

26. Pinto, N., DiCarlo, J.J., Cox, D.D.: Establishing Good Benchmarks and Baselines for Face Recognition. In: IEEE European Conference on Computer Vision, Faces in 'Real-Life' Images Workshop (2008), http://hal.archives-ouvertes.fr/inria-00326732/
27. Pinto, N., DiCarlo, J.J., Cox, D.D.: How far can you get with a modern face recognition test set using only simple features? In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2591–2598. IEEE (2009), http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5206605, doi:10.1109/CVPR.2009.5206605
28. Riesenhuber, M., Poggio, T.: Hierarchical models of object recognition in cortex. Nature Neuroscience 2(11), 1019–1025 (1999), doi:10.1038/14819
29. Serre, T., Kouh, M., Cadieu, C., Knoblich, U., Kreiman, G., Poggio, T.: A Theory of Object Recognition: Computations and Circuits in the Feedforward Path of the Ventral Stream in Primate Visual Cortex. CBCL Paper #259/AI Memo #2005-036 (2005), http://en.scientificcommons.org/21119952
30. Serre, T., Oliva, A., Poggio, T.: A feedforward architecture accounts for rapid categorization. Proceedings of the National Academy of Sciences of the United States of America 104(15), 6424–6429 (2007), http://cat.inist.fr/?aModele=afficheN&cpsidt=18713198
31. Serre, T., Poggio, T.: A neuromorphic approach to computer vision. Communications of the ACM 53(10), 54–61 (2010), http://portal.acm.org/citation.cfm?id=1831425
32. Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., Poggio, T.: Robust Object Recognition with Cortex-Like Mechanisms. IEEE Trans. Pattern Anal. Mach. Intell. 29(3), 411–426 (2007), http://portal.acm.org/citation.cfm?id=1263421&dl=
33. Thorpe, S., Fabre-Thorpe, M.: Seeking categories in the brain. Science 291(5502), 260 (2001), http://www.sciencemag.org/content/291/5502/260.short
34. Thorpe, S., Fize, D., Marlot, C.: Speed of processing in the human visual system. Nature 381(6582), 520–522 (1996), http://www.ncbi.nlm.nih.gov/pubmed/8632824, doi:10.1038/381520a0
35. Turing, A.M.: Computing machinery and intelligence. Mind 59(236), 433–460 (1950)
36. VanRullen, R., Koch, C.: Visual selective behavior can be triggered by a feed-forward process. Journal of Cognitive Neuroscience 15(2), 209–217 (2003), http://www.mitpressjournals.org/doi/abs/10.1162/089892903321208141
37. Wallis, G., Rolls, E.T.: A model of invariant object recognition in the visual system. Progress in Neurobiology 51, 167–194 (1997), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.48.880&rep=rep1&type=pdf
38. Wallisch, P., Movshon, J.: Structure and Function Come Unglued in the Visual Cortex. Neuron 60(2), 195–197 (2008), http://linkinghub.elsevier.com/retrieve/pii/s0896-6273%2808%2900851-9
39. Wersing, H., Körner, E.: Learning optimized features for hierarchical models of invariant object recognition. Neural Computation 15(7), 1559–1588 (2003), http://www.mitpressjournals.org/doi/abs/10.1162/089976603321891800, doi:10.1162/089976603321891800

Actionable Information in Vision

Stefano Soatto

Abstract. A notion of visual information is introduced as the complexity *not* of the raw images, but of the images after the effects of nuisance factors such as viewpoint and illumination are discounted. It is rooted in ideas of J. J. Gibson, and stands in contrast to traditional information as entropy or coding length of the data regardless of its use, and regardless of the nuisance factors affecting it. The non-invertibility of nuisances such as occlusion and quantization induces an "information gap" that can only be bridged by controlling the data acquisition process. Measuring visual information entails early vision operations, tailored to the structure of the nuisances so as to be "lossless" with respect to visual decision and control tasks (as opposed to data transmission and storage tasks implicit in communications theory). The definition of visual information suggests desirable properties that a visual representation should possess to best accomplish vision-based decision and control tasks.

1 Preamble

This paper discusses the role visual perception plays in the "signal-to-symbol barrier" problem.

The "signal-to-symbol barrier" stems from the observation that perceptual agents, from plants to humans, perform measurements of physical processes at a level of granularity that is *essentially continuous*.[1] They also perform actions in the continuum of physical space. And yet, cognitive science, primary epistemics, and in general modern

Stefano Soatto
University of California, Los Angeles, USA
e-mail: soatto@ucla.edu

[1] The continuum is an abstraction, so here "continuous" is to be understood as existing at a level of granularity significantly finer than the resolution of the measurement device or actuator. For instance, although retinal photoreceptors are finite in number, we do not perceive discontinuities due to retinal sampling. Even when the sensory signals and the actions are discrete (e.g., due to digital encoders or transducers), the "analog-to-digital" conversion usually occurs in a manner that is independent of the signal being sampled (*e.g.* fixed-rate sampling), or dependent only on coarse phenomenological aspects of the signal (*e.g.* adaptive sampling based on frequency characteristics or sparsity constraints).

R. Cipolla et al. (Eds.): Machine Learning for Computer Vision, SCI 411, pp. 17–48.
springerlink.com

philosophy, associate "intelligent behavior" with some kind of *internal representation* consisting of discrete symbols ("concepts", "ideas", "objects", "categories") that can be manipulated with the tools of logic or probabilistic inference. But little is known about why such a "signal-to-symbol" conversion should occur, whether it would yield an advantage, or what principles should guide such a discretization process.

Traditional Information Theory, Statistical Decision Theory, and Control Theory shed little light on this process, and indeed suggest that it may be counterproductive. If we consider biological systems as machines that perform actions or make decisions in response to stimuli in a way that maximizes some decision or control objective, then Rao and Blackwell ([65] page 88), or the "Data Processing Inequality," indicate that the best possible agents would avoid "breaking down the data into pieces," *i.e. data analysis*[2], or for that matter any kind of intermediate decision unrelated to the final task, as would instead be necessary to have a discrete internal representation.[3]

So, why would we need, or benefit from, an internal representation? Is "intelligence" not possible in an analog setting? Or is *data analysis* necessary for cognition? If so, what would be the principles that guides it?

And with respect to the academic field of Computer Vision, why have we been performing data analysis (edge detection, feature selection, segmentation, image parsing etc.) against the basic tenets of (traditional) Information and Decision Theory? The latter would suggest that, eventually, a reductionist approach where images are fed raw into a black-box decision or control machine will be most successful. Or perhaps, on the contrary, the traditional notion of Information should be revised, and this revision will point to new principles for data analysis, and validate what Computer Vision scientists have done for decades.

Yet another possibility is that data analysis is not guided by any principle, but an accident due to the constraints imposed by biological hardware, as implied by Turing in [77], where he showed that reaction-diffusion partial differential equations (PDEs) that govern neuronal ion concentrations, although continuous in nature, exhibit discrete solutions. So, if we want to build machines that interact intelligently

[2] Note that I refer to *data analysis* as the process of "breaking down the data into pieces" (cfr. gr. *analyein*), *i.e.* the generally lossy conversion of data into discrete entities. This is not the case for global representations such as Fourier or wavelet decomposition, or principal component analysis (PCA), that are unfortunately often referred to as "analysis" because such techniques were developed in the context of harmonic analysis, a branch of mathematics. The Fourier transform is globally invertible, which implies that there is no loss of data, and PCA consists in linear projections onto subspaces.

[3] Discretization is often advocated on complexity grounds, but complexity calls for data *compression*, not necessarily for data *analysis*. Any complexity cost could be added to the decision or control functional, and the best decision would still avoid data analysis. For instance, to simplify a segment of a radio signal one could represent it as a linear combination of a small number of (high-dimensional) bases, so few numbers (the coefficients) are sufficient to represent it in a parsimonious manner. This is different than breaking down the signal into pieces, *e.g.* partitioning its domain into subsets, as implicit in the process of encoding a visual signal through a population of neurons each with a finite receptive field. So, is there an evolutionary advantage in data analysis, beyond it being just a way to perform lossy data compression?

with their surroundings and are not bound by the constraints of biological hardware, should we draw inspiration from biology, or would it better to jettison it?

The question of representation is ill-posed outside the scope of a task. A task can be as narrow as a binary decision, such as the presence/absence of a person in a scene, or as general as "survival," but in the context of visual perception I distinguish four broad classes of tasks, which I call the four "R's" of vision: Reconstruction (building models of the geometry of the scene), Rendering (building models of the photometry of the scene), Recognition (or, more in general, vision-based decisions such as detection, localization, categorization), and Regulation (or, more in general, vision-based control such as tracking, manipulation etc.).

For Reconstruction and Rendering, I am not aware of any principle that suggests an advantage in data analysis. It is not accidental that the current best approaches to reconstructing models of the geometry and photometry of a scene from image streams recover (piecewise) continuous surfaces and radiance functions directly from the data, as opposed to the traditional multi-step pipeline[4] that was long favored on complexity grounds [53].

In this manuscript, I explore the issue of representation for decision and control tasks. I will avoid defining "intelligent behavior" or even knowledge, other than to postulate that knowledge – whatever it is – *comes from data*, but it is *not data*. This leads to the notion of the "useful portion" of the data, which one might call "information." So, the first step is understanding what "information" means in the context of visual perception. That is the subject of this manuscript.

Fig. 1 The **Sea Squirt**, or Tunicate, is an organism capable of mobility, of predatorial nature, until it finds a suitable rock to cement itself in place. Once it becomes stationary, it digests its own cerebral ganglion, or "eats its own brain" and develops a thick covering, a "tunic" for self defense.

[4] A sequence of "steps" including point-feature selection, wide-baseline matching, epipolar geometry estimation, motion estimation, triangulation, epipolar rectification, dense rematching, surface triangulation, mesh polishing, texture mapping.

What I will show is that visual perception plays a key role in understanding the signal-to-symbol barrier. Specifically, the need to be able to perform decision and control tasks in a manner that is independent of nuisance factors that affect the image formation process leads to an internal representation that is intrinsically *discrete*, and yet *lossless*, in a sense to be made clear soon. However, for this to happen the perceptual agent has to have control over certain aspects of the sensing process. This ties together inextricably sensing and control, in the sense that without the ability to control the sensing process with motion, a discrete internal representation would be a sure loss. A peculiar illustration of this phenomenon is the case of Sea Squirts, or Tunicates, shown in Fig. 1. These are organisms that possess a nervous system (ganglion cells) and the ability to move (they are predators), but eventually settle on a rock, become stationary and thence swallow their own brain.

2 Introduction

More than sixty years ago, Norbert Wiener stormed into his students' office enunciating *"entropy is information!"* before immediately storming out.[5] Claude Shannon later made this idea the centerpiece of his Mathematical Theory of Communication, formalizing and unifying the wide variety of methods that practitioners had been using to transmit signals through channels. The influence of Shannon's communication theory has since spread beyond the transmission and compression of data, and is now broadly known as Information Theory. But is the entropy of the *data* really "information"? There is no doubt that the more complex the data, the more costly it is to *store* and *transmit*. But what if we want to *use* the data for *tasks* other than storage or transmission? What is the "information" that an image contains about the *scene* it portrays? What is the value of an image if we are to *recognize* objects in the scene, or *navigate* through it? (Fig. 2).

Despite its pervasive reach today, Shannon's notion of information had early critics,[6] among those James J. Gibson, who wrote *"My theory of the available information in ambient light is radically different from [that of] Shannon. [...] My notion is that information consists of invariants underlying change"* [30].[7] Already in the fifties he was convinced that *data* is not *information*,[8] and the value of data should depend on what one can do with it, *i.e.* the task [55]. Much of the complexity in

[5] R. Fano, personal communication.

[6] Even Shannon's disciples acknowledge that "Information Theory is a total misnomer [...] it does not deal with information at all, it deals with data" (R. Gallagher, personal communication).

[7] It is only unfortunate that, in engineering communications, the signals are heavily structured so the nuisance, often dubbed "noise," is usually assumed to be additive, zero-mean, white, and Gaussian. As a consequence, the issue of invariance was never thoroughly explored, although, interestingly, Wiener was aware of it. In ([82], p. 50) he even introduced the first (integral) moment as an invariant statistic to a group (eq. (6.01), p. 138) and called it a *gestalt*!

[8] He was nevertheless rooted in empirical epistemology and therefore assumed that *information comes from data*.

an image is due to *nuisance factors*, such as illumination, viewpoint and clutter,[9] that have little to do with the decision (perception) and control (action) task at hand. So it is intuitive that the value of data should relate to its complexity *only after the effects of nuisance factors has been discounted*.[10] Unfortunately, any constant function is an "invariant underlying change", so Gibson was missing the other facet of information that relates to its "usefulness" (sufficiency) towards the task.

The goal of this manuscript is to define an operational notion of "information" that is relevant to visual inference tasks, as opposed to the transmission and storage of image data. Following Gibson's lead, I define *Actionable Information* to be the complexity of a maximal statistic that is invariant to the nuisances associated to a given task. According to this definition, the Actionable Information in an image depends *not just* on the complexity of the data, but also on the *structure* of the scene it portrays. I illustrate this on a simple environmental exploration task, that is central to Gibson's ecological approach to perception. A robot seeking to maximize Shannon's information (a "Shannonian Explorer") is drifting along unaware of the structure of the environment, while one seeking to maximize Actionable Information (a "Gibsonian Explorer") is driven by the topology of the surrounding space. Both measure the same *data* (images), but the second is *using it*

[9] I use the word "nuisance" in the standard sense of statistical inference; this does not imply that nuisance factors are dismissed or irrelevant. It just means that they affect the data, but not the task. Gibson wrote: *"Four kinds of invariants have been postulated: those that underlie change of illumination, those that underlie change of the point of observation, those that underlie overlapping samples, and those that underlie a local disturbance of structure. [...] Invariants of optical structure under changing illumination [...] are not yet known, but they almost certainly involve ratios of intensity and color among parts of the array. [...] Invariants [...] under change of the point of observation [...] some of the changes [...] are transformations of its nested forms, but [...] The major changes are gain and loss of form, that is, increments and decrement of structures, as surfaces undergo occlusion." [...] The theory of the extracting of invariants by a visual system takes the place of theories of "constancy" in perception, that is, explanations of how an observer might perceive the true color, size, shape, motion and direction-from-here of objects despite the wildly fluctuating sensory impressions on which the perceptions are based."* ([31], p. 310).

[10] Appealing as the idea of characterizing "invariants under change" sounds in words, a modern Computer Vision scientist would dismiss it at the outset, for it has since become known that such invariants *do not exist*. Invariants were considered "the Holy Grail of Computer Vision" in the eighties, until [16] and [21] showed that they do not exist neither for viewpoint, nor for illumination. Lacking mathematical and computational foundations that enable engineering applications, Gibson's ideas thus remain confined to the realm of philosophy [39] and perception psychology. However, recent developments have shown that the situation is more complex than commonly assumed: [16] refers to statistics of *point features,* not *images,* and [78] shows instead that non-trivial viewpoint invariants always exist for images of Lambertian objects of general shape. Similarly, [21] consider general illumination fields, but invariants can be constructed for simpler illumination models, such as contrast transformations [2], even though these are valid only locally. Invariance always refers to an underlying model, that is as good as the assumptions it is based on, and as useful as the ensuing algorithms are for the task of interest. Invariance to even more restricted classes of transformations is the underpinning of very simple image statistics that have recently gained significant popularity in visual recognition and categorization tasks.

to accomplish spatial tasks.[11] This manuscript relates to work in information theory, video compression, robotics, visual recognition, as I discuss in Sect. 7. There, I also discuss the visual representations that our operative definition implies as "lossless" relative visual decision or control tasks.

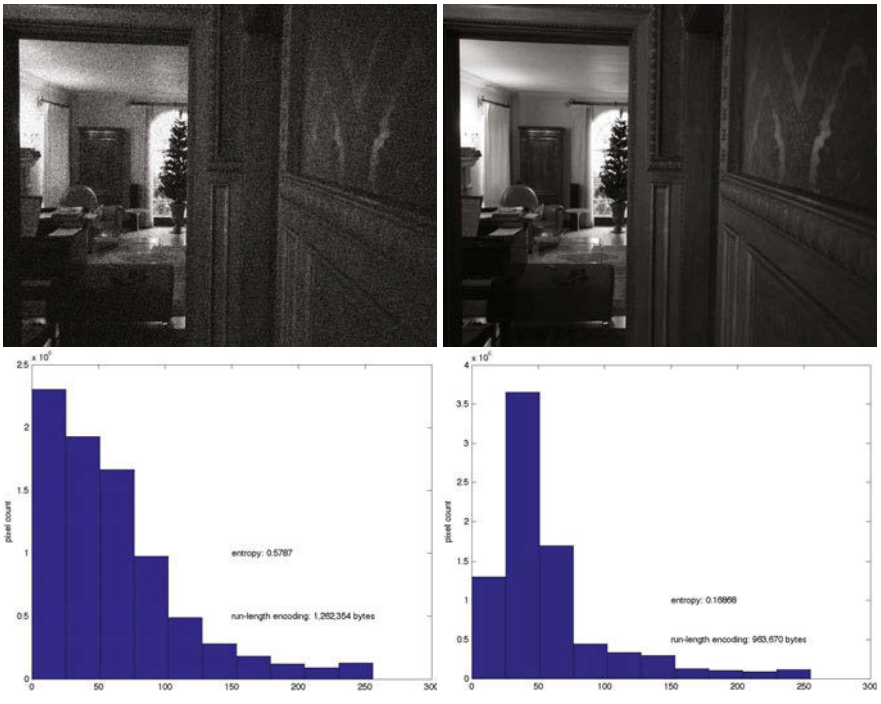


Fig. 2 The left image is more costly to store or transmit than the right image. However, our goal is to *use* these images for a decision or control task that involves properties of the *scene*.

3 Preliminaries

The paper is structured in the following way.

- In the previous section, as a way of motivation, I have argued that traditional information theory, as developed with an eye towards the problem of

[11] It could be argued that Shannon would not seek to maximize the entropy of the data, but instead the mutual information between the scene and the data (which he called "equivocation"). Our exercise can be thought of a way to formalize this notion, but avoiding having to place an explicit probability distribution on the set of scenes, which is a tall order. Furthermore, while it is easy to formally define the mutual information between the scene and the image, computing it for an erasure channel (occlusions) under compositional infinite-dimensional domain warpings (viewpoint changes) and multiplicative infinite-dimensional disturbances (illumination) is not something easily done using the tools developed in classical Information Theory.

"reproducing" the output of a source, is inadequate to characterize the value of an individual image for the purpose of *decision* or *control* tasks relative to the scene that the image portrays. Images are affected by *"nuisance factors"* that act on the data in a complex and highly structured fashion. Although closer to our scope, Gibson's notion of information as "invariants under change" falls short because it does not consider the counterpart of invariance, which is the *"discriminative"* (decision) or *"reachable"* (control) component of the representation.

- In Sect. 4.1, I introduce the notion of *"actionable information"* as the complexity of the maximal statistic that is invariant to a given nuisance. Similarly, I define *"complete information"* as the minimal statistic that is sufficient for a given task. When the nuisance is *"invertible"*, the two are identical, and their difference, the *"actionable information gap"* defined in Sect. 4.3, is zero.
- In the context of vision, viewpoint and illumination – away from visibility artifacts such as occlusions and cast shadows – are *invertible.* However, occlusions, cast shadows and quantization are *not.* Therefore, in general, the actionable information gap is non-zero.[12]
- The invertibility of a nuisance depends on the *control authority* of the sensor. While *occlusions and quantization* are non-invertible for a passive and static observer, they *become invertible when the observer is able to control the data acquisition process* (Sect. 4.3) for instance by changing viewpoint or accommodation (Fig. 10). Similarly, cast shadows are not invertible in grayscale images, but may become invertible when one can sample multiple spectral bands. The process of "information pickup" consists in the exploration of the environment aimed at closing the information gap (Sect. 4.4).
- While complete information, in general, cannot be measured, actionable information can be computed. I describe a representational structure that organizes two-dimensional (region statistics), one-dimensional (boundaries) and zero-dimensional (attributed points) image characteristics at all scales; its coding length measures actionable information (Sect. 5). This is a conceptual construction. Nevertheless, a "poor man's version" of this construction can be easily and efficiently computed.
- Since complete information, and therefore the actionable information gap, cannot be known in advance, perceptual exploration must proceed based on locally computable quantities. I define the Actionable Information Increment as a quantity that can be computed instantaneously (in the presence of infinitesimal motion) and integrated over time as part of the exploration process (Sect. 4.3).
- Finally, in Sect. 7 I discuss some consequences of these arguments, including a suggested set of prescriptions that a visual representation should obey, and their effects on the "signal-to-symbol barrier."

Before articulating our arguments, we need to introduce certain definitions, for which we need some notation.

[12] Indeed, it is infinite, for actionable information is zero (there is no occlusion-invariant in one image) and the complete information is infinity (to compute a sufficient statistic with respect to occlusion one would have to acquire the entire light field).

3.1 Notation and Conventions

An image is represented as a function $I : D \subset \mathbb{R}^2 \to \mathbb{R}^k_+;\ x \mapsto I(x)$ that is $\mathbb{L}^2$-integrable, but otherwise not necessarily continuous, taking positive values in k spectral bands, *e.g.* $k = 3$ for color, and $k = 1$ for grayscale. A time-indexed image is indicated by $I(x,t)$, $t \in \mathbb{Z}_+$, assuming a discrete temporal sampling, and a sequence is denoted by $\{I(x,t)\}_{t=1}^{T}$, or simply $\{I\}$. The image relates to the scene, which is represented as a collection of piecewise continuous surfaces ("shape") $S \subset \mathbb{R}^3$, possibly parameterized by x, $S : D \to \mathbb{R}^3;\ x \mapsto S(x)$, and a reflectance $\rho : S \to \mathbb{R}^k$, which is also parameterized, with an abuse of notation exploiting visibility constraints, as $\rho(x) \doteq \rho(S(x))$. I indicate points in space with capital letters $X \in \mathbb{R}^3$, and points in the image with $x \in \mathbb{R}^2$. I model illumination changes by contrast transformations, *i.e.* monotonically increasing continuous functions $h : \mathbb{R}^+ \to \mathbb{R}^+$. This is a rough approximation for Lambertian objects viewed under ambient illumination, where the radiance ρ corresponds to the diffuse albedo. Changes of viewpoint are rigid body transformations, *i.e.* elements of the Special Euclidean group $g \in SE(3)$, represented by a translation vector $T \in \mathbb{R}^3$ and a rotation matrix $R \in SO(3)$, indicated by $g = (R,T)$ [53]. As a result of a viewpoint change, points in the image domain $x \in D$ are transformed (warped) via $x \mapsto \pi(g^{-1}(\pi_S^{-1}(x))) \doteq w(x)$, where $\pi : \mathbb{R}^3 \to \mathbb{P}^2;\ X \mapsto \bar{x} = \lambda X$ is an ideal perspective projection and $\lambda^{-1} = [0\ 0\ 1]X \in \mathbb{R}_+$ is the depth along the projection ray $[x] \doteq \{X \in \mathbb{R}^3 \mid \exists\ \lambda \in \mathbb{R}_+, x = \lambda X\}$; π_S^{-1} is the inverse projection, that is the point of first intersection of the projection ray $[x]$ with the scene S. I use the notation $w(x;S,g)$ when emphasizing the dependency of w on viewpoint and shape. Without loss of generality [72], I represent changes of viewpoint with diffeomorphic domain deformations $w : D \subset \mathbb{R}^2 \to \mathbb{R}^2$. This model is viable only away from visibility artifacts (occlusions, cast shadows), which are discussed in Sect. 3.2. All un-modeled phenomena (deviation from Lambertian reflection, complex illumination effects etc.) are lumped into an additive "noise" term $n : \mathbb{R}^2 \to \mathbb{R}^k$. We finally have our image formation model:

$$\begin{cases} I(x) = h(\rho(X)) + n(x) \\ x = \pi(g(X)),\ \ X \in S. \end{cases} \tag{1}$$

Summary: (Refer to Fig. 3) I call the image I, the reflectance ρ, illumination (contrast) h, warping w, which depends on the shape S and the viewpoint g. I further call the scene ξ, the collection of (three-dimensional, 3D) shape and reflectance $\xi \doteq \{\rho, S\}$, and the nuisance ν, the collection of viewpoint and illumination $\nu \doteq \{g,h\}$. In short-hand notation, substituting X in the first equation above with $g^{-1}(\pi_S^{-1}(x))$, I write (1) as

$$\boxed{I(x) = h \circ \rho \circ w(x;S,g) + n(x) \doteq f(\rho, S; g, h, n)} \tag{2}$$

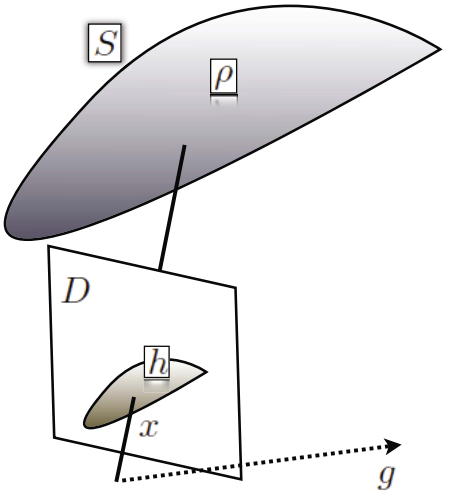


Fig. 3

or, again with an abuse of notation, as

$$\boxed{I = f(\xi; \nu).}$$

3.2 Visibility and Quantization

The model (1) is only valid away from visibility artifacts such as occlusions and cast shadows. I will not deal with cast shadows, and assume that they are either detected from the multiple spectral channels $k \geq 3$, or that illumination is constant and therefore they cannot be told apart from material transitions (*i.e.* in the reflectance ρ). Occlusions, on the other hand, we cannot do away with. Based on empirical studies of natural intensity and range statistics [36, 57], I model occlusions as the "replacement" of f, in a portion of the domain[13] $\Omega \subset D$, by another function β having the same statistics [57].[14] Sometimes Ω is called the *background* even though, in practice, it can be in front of the object of interest, or it can be part of the object of interest itself, as in self-occlusions:

$$I(x) = \begin{cases} f(\rho, S; g, h, n) & x \in D \backslash \Omega \\ \beta(x) & x \in \Omega. \end{cases} \tag{3}$$

[13] Note that Ω is not necessarily simply connected, and this model does not impose restrictions on how many depth layers there can be [5].

[14] One cannot distinguish an occluding boundary from a material transition in a single pinhole image, unless the sensing process enables changes of viewpoint or accommodation (Fig. 10).

Digital images are spatially quantized into a discrete lattice, with each element averaging the function I over a small region $\mathcal{B}_v(x_{ij}) \subset D$ of size $v \in \mathbb{R}_+$ centered at $x_{ij} = (iv, jv),\ i, j \in \mathbb{Z}$:

$$I(i,j) = \int_{\mathcal{B}_v(x_{ij})} I(x)dx = I(x_{ij}) + n(x_{ij}) \tag{4}$$

where the quantization error is lumped into the additive noise n. In what follows, depending on the context, we may lump occlusions Ω, β, quantization and noise n among the nuisances ν.

3.3 Invariant and Sufficient Statistics

A statistic, or "feature," is a deterministic function ϕ of the data $\{I(x), x \in D\}$, taking values in some vector space, $\phi(I) \in \mathbb{R}^K$. I indicate this in short-hand notation via $\phi(I)$. A statistic is *invariant* if its value does not depend to the nuisance, *i.e.* for any $\nu, \bar{\nu}$, we have $\phi(f(\xi, \nu)) = \phi(f(\xi, \bar{\nu}))$. A trivial example of invariant feature is a constant function $\phi(I) = c\ \forall\ I$. Among all invariant statistics, we are most interested in the *largest*[15], also called *maximal invariant*

$$\boxed{\hat{\phi}(I).}$$

A statistic is *sufficient* for a particular *task*, specified by a risk functional R associated to a control or decision policy u and loss function L, $R(u|I) \doteq \int L(u, \bar{u})dP(\bar{u}|I)$, $R(u) \doteq \int R(u|I)dP(I)$, if the risk based on a policy computed using such a statistic is the same as the risk based on the raw data, *i.e.* $R(u|I) = R(u|\phi(I))$. A trivial example of sufficient statistic is the identity $\phi(I) = I$. Among all sufficient statistics, of particular interest is the smallest, or *minimal*, one

$$\boxed{\phi^{\vee}(I).}$$

Note that, in general, $R(u|I) \leq R(u|\phi(I))$ for any measurable function ϕ (the *"Data Processing Inequality"*), with equality defining ϕ as a sufficient statistic. When a nuisance acts as a *group* on the data, it is always possible to construct invariant sufficient statistics (the orbits, or the quotient of the data under the group). In that case, the policy u is called an *equivariant* classifier (for decisions) or controller (for actions) ([65], Theorem 7.4). It would be possible, as an alternative, to define sufficient statistics in terms of Fisher's Information.

[15] Maximal in the sense of inclusion of sigma-algebras.

4 Placing the Ecological Approach to Visual Perception onto Computational Grounds

4.1 Actionable Information

I define *Actionable Information* to be the complexity[16] H of a maximal invariant,

$$\boxed{\mathcal{H}(I) \doteq H(\hat{\phi}(I)).} \tag{5}$$

When the maximal invariant is also a sufficient statistic, we have the *complete information*

$$\mathcal{I} \doteq H(\phi^{\vee}(I)) = \mathcal{H}(I). \tag{6}$$

In this case, the Actionable Information measures all and only the portion of the data that is relevant to the task, and discounts the complexity in the data due to the nuisances. As is discussed in Sect. 4.3, invariant and sufficient statistics are, in general, different sets, so we have an *"information gap."* In Sect. 5 I show how to compute Actionable Information.

In the next section I show that for some nuisances (invertible), the gap can be reduced to zero, whereas for other nuisances (non-invertible), the gap can be infinite.

4.2 Invertible and Non-invertible Nuisances

Viewpoint g and contrast h act on the image as *groups*, in the absence of occlusions and cast shadows, and therefore can be *inverted* [72]. In other words, the effects of a viewpoint and contrast change, away from visibility artifacts, can be "neutralized" in a single image, and an invariant sufficient statistic can, at least in principle, be computed [72]. Note that the notion of sufficient statistic in this case is with respect to any distribution, since it is possible to reconstruct an individual realization of the scene regardless of the nuisance. Fig. 5 illustrates this, and [2] and [78] prove it for contrast and viewpoint respectively. It may be puzzling that the statistics that are invariant to contrast (the geometry of the iso-contours of the image [19]) are not invariant to viewpoint, and those that are invariant to viewpoint (the intensity of the image warped onto a canonical domain [78]) are not invariant to contrast. The conundrum was recently solved in [72] where it was shown that the Attributed Reeb Tree (ART) of a (portion of an) image is the viewpoint-contrast invariant sufficient statistic. The ART stores the label (maximum, minimum, saddle), relative ordering and connectivity of extrema of the function f in (3) and is supported on a zero-measure subset of the image domain, so it is somewhat surprising that this

[16] Complexity is computed relative to a distribution (entropy) or a code (coding length). For now, we will leave the description of the underlying distribution or code vague, as the distribution will be constructed as part of the exploration process. In the meantime, the user can imagine any distribution induced on the maximal invariant by a distribution for the raw data, formally indicated by $p(I)$.

"thin" object is actually equivalent to the entire image but for the effects of viewpoint and contrast changes. If these were the only nuisances, we would be in business. Unfortunately, this is of little help, as *occlusion and quantization are not groups,* and once composed with changes of viewpoint and contrast, the composition cannot be inverted. Or can they? I will address this issue in Sect. 4.3, but not before I have described how to compute viewpoint and illumination invariants that are non-committal with respect to visibility and quantization.

When a nuisance transformation is not a group, its effects cannot be eliminated via pre-processing, and instead must be dealt with as part of the decision or control process: The risk functional R depends on the nuisance, $R(u|f(\xi;\nu))$, which can be eliminated either by extremization, $\max_\nu R(u|f(\xi;\nu))$ following a maximum-likelihood (ML) aproach, or by marginalization $\int R(u|f(\xi,\nu))dP(\nu)$, following a maximum a-posterior (MAP[17]) approach, if a probability measure on the nuisance $dP(\nu)$ is available.[18] In either case, the decision should not be based on direct comparison of two invariant statistics, $\phi(I_1) = \phi(I_2)$ computed separately on the training/template data I_1 and on the testing/target data I_2 in a pre-processing stage. Instead, a costly optimization (ML) or marginalization (MAP) is necessarily performed at run-time. The most one can hope from pre-processing is to pre-compute as much of the optimization or marginalization functional as possible.

Segmentation as Redundant Lossless Coding

An occlusion $\Omega \subset D$, $\beta : \Omega \to \mathbb{R}^k$ is a region that exhibits the same (piece-wise spatially stationary [57]) statistics of the unoccluded scene (3). It can be multiply-connected, generally has piecewise smooth boundaries. Even if we could detect discontinuities in the image, which is a tall order, we would still not know which are the occluding boundaries, as opposed to material transitions or cast shadows. Furthermore, we do not know the statistics of the occluder region, as different quantization scales can cause image structures (extrema and discontinuities) to appear and disappear. Fig. 5 illustrates this phenomenon. In the absence of quantization and noise, one would simply detect all possible discontinuities, store the entire set $\{f(\xi,\nu),\ \forall\ \nu\}$, leaving the last decision bit (occlusion vs. material or illumination boundary) to the last stage of the decision or control process, performed either by extremization (ML) or marginalization (MAP). Occluders connecting to the ground (such as the tree in the "Flower Garden" sequence [23]) where no occlusion boundary is present would have to be "completed" as advocated by Gestalt psychologists [80], leading to a *segmentation*, or partitioning, of the image domain into regions with smooth statistics. Unfortunately, quantized signals are everywhere discontinuous, making the otherwise

[17] Invariant classification is problematic in a Bayesian setting, as one has to use improper uninformative priors; the issue is discussed at length in [63].

[18] Consider for example the binary decision of whether two images I_1 (training, or template image) and I_2 (testing, or target image) portray the same scene ξ. If the nuisance ν involves occlusions, so that $I_1 = f(\xi;\nu_1)$ and $I_2 = f(\xi;\nu_2)$, a decision can be performed by "searching" for all possible scenes ξ and occlusions ν_1, ν_2 that generate both images to within a specified accuracy (threshold). This is equivalent to implicitly "reconstructing" the scene ξ and "registering" the nuisances ν_1, ν_2.

trivial detection of discontinuities all but impossible. One could salvage this approach by setting up a cost functional (a statistic) $\psi_\Omega(I)$, that implicitly defines a notion of "discrete continuity" within Ω but not across its boundary, making the problem of segmentation self-referential (*i.e.* defined by its solution). But while no single segmentation is "right" or "wrong," the set of *all possible segmentations*, defined for *all possible quantization scales*, may be *useful.* It does not reduce the complexity of the image (in fact, it is highly redundant), but it may reduce the run-time cost of the decision or control task, by rendering it a choice of regions and scales that match across images. In Sect. 5 I show how to compute actionable information based on a scale-space segmentation tree.

For any scale $s \in \mathbb{R}_+$, minimizing $\psi_\Omega(I|s)$ yields a different segmentation $\Omega(s) \doteq \arg\min_\Omega \psi_\Omega(I|s)$. Because image "structures" (extrema and discontinuities) can appear and disappear at the same location at different scales,[19] one would have to store the entire continuum $\{\Omega(s)\}_{s\in\mathbb{R}_+}$. In practice, $\psi_\Omega(\cdot|s)$ will have multiple extrema *(critical scales)* that can be stored in lieu of the entire scale-space. This is different than (single) scale selection, as advocated in the scale-space literature. In between such critical scales, structures become part of aggregate statistics that are called *textures* [69]. See Fig. 5. In Sect. 5 I show how to use a (multi-scale) texture segmentation algorithm to compute actionable information.

As described in Sect. 4.1, *in general one cannot compute statistics that are at the same time invariant and sufficient, because occlusion and quantization nuisances are not invertible.*

4.3 The Actionable Information Gap

As I have hinted at in Sect. 3.2, whether a nuisance is invertible depends on the image formation process: Cast shadows are detectable if one has access to different spectral bands. Similarly, occluding boundaries cannot be detected from a single image captured with a pin-hole camera, but they can be detected if one can control accommodation or vantage point. So, if the sensing process involves *control* of the sensing platform (for instance accommodation and viewpoint), then both occlusion and quantization become *invertible* nuisances.[20] This simple observation is the key to Gibson's approach to ecological perception, whereby *"the occluded becomes unoccluded"* in the process of "Information Pickup" [29].

To make this concrete, recall from Sect. 4.1 the definition of *complete information* and note that – because of the non-invertible action of the nuisances – it must now depend[21] on the *scene* ξ. Specifically, given a collection of images $\{I\}$, I call

[19] Two-dimensional signals do not obey the "causality principle" of one-dimensional scale-space, whereby structure cannot be created with increasing scale [51].

[20] Want to remove the effect of an occlusion? Move around it. Want to resolve the fine-structure of a texture, removing the effects of quantization? Move closer.

[21] Actionable information also depends on the scene ξ, but only through the image $I = f(\xi, v)$. Complete information, on the other hand, depends on the scene in ways that are independent of the measured image I.

a *representation any* statistic from which the given data can be generated, up to a residual noise that is statistically simple (homoscedastic, isotropic and temporally white). Of all such statistics, we want to consider, again, the smallest, $\phi_\xi^\vee(\{I\})$. As more and more images are gathered, the set of representations that are compatible with (*i.e.,* that can generate) them becomes smaller. In the limit, when one considers the entire *light field*, that is the set of all possible images that the given scene ξ can generate (that also relates to the so-called *Plenoptic Function*), the resulting representation is called *complete*:

$$H(\phi_\xi^\vee(I)) \geq H(\phi_\xi^\vee(I)|I(x,0)) \geq H(\phi_\xi^\vee(I)|I(x,0),I(x,1)) \geq \ldots$$
$$\ldots \geq H(\phi_\xi^\vee(I)|\{I(x,t)\}_{t=0}^{\infty}) = 0. \quad (7)$$

We can then call the *complete information* the complexity of a complete representation:

$$\boxed{\mathcal{I} \doteq H(\phi^\vee(\{I\})).}$$

Note that, although it may seem impossible or irrelevant to attempt to capture the complexity of the light field, there are indeed computational approaches to measure it [24]. However, in practice the complete information can be unbounded. In either case, unless the nuisances are invertible, it is different from the Actionable Information, so we can define their difference as the Actionable Information Gap (AIG):

$$\boxed{\mathcal{G}(I) \doteq \mathcal{I} - \mathcal{H}(I).} \quad (8)$$

The process of Information Pickup, therefore, is one of reducing the AIG. In general, this process may be asymptotic, although in some cases (*i.e.,* for some tasks) *sufficient exploration* may be accomplished in finite time [62]. Note that, in the presence of occlusion and quantization, *the gap can only be reduced by moving within the environment.* In order to move, however, the agent must be able to compute the effects of its motion on the AIG, ideally without having to know the complete information $\mathcal{I}$, even if the data $\{I\}$ or the statistics $\phi^\vee(\{I\})$ were available from memory of previous explorations.

To this end, we must be able to quantify the *"gain"* that comes from taking additional measurements in response to a control action. This is the process of visual exploration, which we describe in the next subsection.

It is clear that occlusions play a critical role in the process: As we move in any non-trivial environment, portions of the scene that were previously occluded become visible, and the Actionable Information computed on the subset of the image that has become unoccluded constitutes one portion of the "Actionable Information Increment." Therefore, occlusion detection is central to visual exploration. We refer the reader to [5] for a description of a method for occlusion detection that is based

on the convex relaxation of a variational optimization problem, that admits a global solution with respect to both the motion field w and the occluded domain Ω.[22]

4.4 Information Pickup

To study the process of "Information Pickup" by means of closing the Actionable Information Gap, I specify a simple model of an "agent," that is a Euclidean reference frame in physical space, *i.e.* a *viewpoint* $g(t) \in SE(3)$, moving under the action of a control, which I assume can specify the instantaneous velocity $u(t) \in \mathbb{R}^6$. This kinematic model neglects masses, inertias and other dynamic fancy. The agent simply moves by integrating its velocity, *i.e.* the ordinary differential equation $\dot{g}(t) = \widehat{u}(t)g(t)$ starting from some initial position, which for convenience I assume to be the origin $g(0) = e$.[23] The agent measures an image at each instant of time, $I(x,t)$:

$$\boxed{\begin{cases} \dot{g}(t) = \widehat{u}(t)g(t) \quad g(0) = e \\ I(x,t) = f(\rho(x), S(x); g(t), h(t), n(x,t)). \end{cases}} \tag{9}$$

We call a system constrained by the dynamical model above, aiming to minimize the AIG, an *explorer*. A myopic explorer would simply try to maximize, at each instant of time, the *Actionable Information Increment* (AIN):

$$\hat{u}(t) = \arg\max_{u} \mathcal{H}(I(x,t)|I(x,t-1)) \;\text{ subject to (9)} \tag{10}$$

where the function being minimized can be computed once occlusions are detected as described in the previous subsection. In addition to being myopic, such an explorer would be *memoryless*, so it would continue exploring portions of the scene it has seen in the past, just not the immediate past. To remedy that, one could design an explorer with memory, by maximizing instead $\mathcal{H}(I(x,t)|\{I(x,\tau)\}_{\tau=0}^{t-1})$. Clearly, such an explorer would be difficult to maintain as its memory grows unbounded. Therefore, one could condition, rather than on the past history, $\{I(x,\tau)\}_{\tau=0}^{t-1}$, on a statistic, call it $\hat{\xi}(t)$, that is inferred incrementally as the minimal sufficient statistic of the past history that can generate the history up to a statistically simple residual. But this is precisely what we defined as a *representation* earlier, only that it is not complete for any finite time $t < \infty$. Instead, we can construct $\hat{\xi}(t)$ in such a way that it will converge to a complete representation, by minimizing

[22] A simple approximate solution can be found by block matching followed by run-length encoding of the residual, as customary in MPEG. Efficient algorithms, including hardware implementations, are readily available for this task. The shortcoming of this approach is that, in general, it yields a loss of actionable information, whereas the optimal solution guarantees, at least in theory, that no actionable information is lost.

[23] Here $\widehat{u}$ indicates the operator that transforms a linear and angular velocity vector u into a *twist* $\widehat{u} \in se(3)$ [53].

$$\hat{\xi}(t+1) = \arg\min \mathcal{H}(I(x,t+1)|\hat{\xi}(t)) \tag{11}$$

in conjunction with the maximization of the Actionable Information Increment with respect to the control action

$$\hat{u}(t) = \arg\max_{u} \mathcal{H}(I(x,t)|\hat{\xi}(t)) \;\text{ subject to (9)}. \tag{12}$$

The process could be initialized with $\hat{\xi}(0) = \{\pi_0, I(x,0)\}$, where π_0 is the surface corresponding to the image plane, and $I(x,0)$ is the first measured image. In other words, initially the representation is simply an image (or its maximal invariant) attached to a plane. Note that a representation is a subtle object, because it is a function of the images (it is a statistic) but it lives in the embedding space of the space of scenes (so it can be thought of as a scene); it can be used to generate images, and these images can be compared with those generated by the actual ("true") scene. We will never be able, however, to determine whether the representation is close to the "true scene." What we *can* do is to determine whether the images generated by the representation (or their maximal invariants) are close to images generated by the actual scene. This is convenient because it avoid philosophical as well as computational entanglements, because we do not need to put a measure in the space of scenes, which is a notoriously hairy thing to do.

A more sophisticated (non-myopic) explorer would try to plan its control based on a receding horizon of (possibly infinite) length T,

$$\hat{u}(t) = \arg\sup_{u(\cdot)} \mathcal{H}(\{I(x,\tau)\}_{\tau=t}^{T}|\xi(t),u) \quad \text{subject to (9)}. \tag{13}$$

This would achieve exploration more efficiently than a myopic strategy, although if efficiency is not a priority, a random walk (say the Roomba vacuum cleaner) will eventually explore the unknown space (see Sect. 6). The process is depicted in Fig. 4.

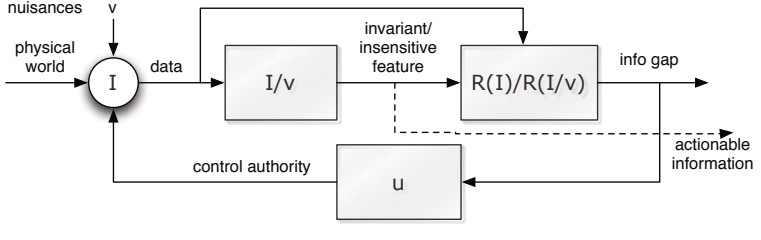


Fig. 4 For a given task, represented by a risk functional R, and for a given nuisance v, one can in general compute invariant statistics. Their value is determined by the information gap: When it reaches zero, the invariant is also sufficient for the task. When it is larger, control authority can be exercised to minimize it; thence the invariant/insensitive feature is also sufficient (dashed line).

Remark 1 (The Actionable Information Paradox). *Consider the task of recognizing an object that can exhibit significant reflectance variability, such as chameleon, or a passenger vehicle on the road. What determines the identity of the object is its three-dimensional shape. Naturally one wants a representation of shape that is viewpoint-invariant. But viewpoint cannot be "undone" from an image alone, so one would have to store the entire image and defer dealing with viewpoint as part of the matching process. If, however, one moves relative to the object of interest, then three-dimensional shape is observable, and one can infer, and store, a 3D model of the geometry of the object, and discard photometry (reflectance and illumination), thus effectively reducing the storage requirement to little more than the size of a single image (assuming piecewise smooth surfaces). This yields the apparent paradox whereby more data yield a smaller storage requirement. It also means that, in order to "extract information" we have to "throw away some of the data," which has epistemological implications discussed in Sect. 7.*

5 Representational Structures

I now describe the computation of Actionable Information and the representation it dictates.

5.1 Computing Actionable Information

For each image, we first compute a viewpoint-contrast invariant as follows: First, we perform (over-) segmentation at all possible scales: Starting from a 5-dimensional vector of color channels and positions, I use Quick Shift [79] to construct in one shot the tree of all possible segmentations (Fig. 6 top). I then consider the finest partition (a.k.a. "superpixels") to be the elementary unit, and construct the adjacency graph, then aggregate nodes based on the histogram of vector-quantized intensity levels and gradient directions in a region ω of 8×8 pixels and arrive at the *texture adjacency graph* (TAG) (Fig. 7 top-right). Two-dimensional regions with homogeneous texture (or color) are represented as nodes in the TAG. I then represent one-dimensional boundaries between texture regions as edges in the TAG, or equivalently pairs of nodes (Fig. 6 top-right and Fig. 7 bottom-left). Ridges sometimes appear as boundaries between textured regions, or as elongated superpixels. Finally, I represent zero-dimensional structures, such as junctions or blobs (Fig. 6 bottom), as faces of the TAG, or equivalently pairs of edges (Fig. 7 bottom). This structure is the *Representational Graph*, $\mathcal{R}$, whose run-length encoding measures Actionable Information. In particular, $\mathcal{H}(I)$ is computed by summing, over the number of nodes $N(s)$ of the representational graph over all stored scales s, the coding length of the texture histograms associated to each node, but not the shape or size of the regions ω_i; the strength associated to each edge, corresponding to the probability of detection of an edge and ridge detector and the proximity to a superpixel boundary, but not the shape or length of the boundary, using our implementation of [51] (Fig. 6

top-right); the presence of an attributed point region associated to a face and its descriptor, but not the position of the point. I use a SIFT detector for Difference-of-Gaussian blobs from VLfeat (http://www.vlfeat.org), and Harris-Affine from [56] (Fig. 6 bottom-right and bottom-left respectively). Although in theory we should also store, for each of these regions, the ART [72], in practice, in the experiments reported in Sect. 6, I forgo this step.

This construction is conceptual; in practice, all that matters during the exploration phase is that the information gap is being closed, possibly in an efficient manner.

5.2 Closing the Actionable Information Gap

To compute a representation one can start with a planar surface where the first image is supported, and compute the representational structure for that image. From the second image onward, one can compute occlusions *relative to the representation* (note that the representation can be thought of as a *scene*), and add nodes and edges to the representational graph depending on what structures were revealed by the new image. This process is conceptually equivalent of performing on-line causal reconstruction of shape and reflectance [40]. However, rather than dense Euclidean geometry and albedo, we only store a multi-scale discrete representation capturing the topology of the space as well as a contrast-invariant local descriptor of the albedo [42].

In order to compute the AIG, the complete information is necessary. This is, in general, not available unless one has had the opportunity to inspect the environment beforehand, and has, for instance, the entire light-field stored in memory. The process of Information Pickup hinges on the hypothesis that, by integrating the AIN, one would eventually converge to complete information, hence rendering the AIG equal to zero. In the next section I validate this assumption empirically.

An explorer capable of making the AIG (asymptotically) zero could be called an *omnipotent* explorer. In practice, any controller will have some limitations, and therefore one can expect that there would be a tradeoff between the *control authority* of the explorer (to be properly defined) and the limit of the AIG. This is discussed in [70].

6 Empirical Consequences of the Definitions

In this section I test our hypothesis that an agent guided by Gibson would seek to "go around occlusions" and "resolve textures," whereas one guided by Shannon would be unaware of the topological structure of the environment, despite using the same *data*.

In the first indoor experiment (Fig. 8), a simulated robot is given limited control authority $u = [u_X,\ u_Y,\ 0,\ 0,\ 0,\ 0]^T$ to translate on a plane inside a (real) room, while capturing (real) color images with fixed heading and a field-of-view

of 90^o. The robot is capable of computing both Entropy and the AIN at the current position as well as at immediately neighboring ones. Under these conditions, the agent reduces to a point (the vantage point) $g = (Id, T)$ where $T = [T_X, T_y, 0]^T$. I indicate the vantage point with $X = [T_X, T_Y]^T$, consistent with the nomenclature introduced in Sect. 3.1, and the control with $V = [u_X,\ u_Y]^T$.

In the second outdoor experiment (Fig. 11), the robot is Google's StretView car,[24] over which we have no independent control authority. Instead, I assume that it has an intelligent (Gibsonian) driver aboard, who has selected a path close to the optimal one. The robot measures omnidirectional panoramas at each instant of time, so the data is symmetric with respect to forward or backward traversal. In this case, we cannot test independent control strategies. Nevertheless, we can still test the hypothesis that traditional information, computed throughout the sequence, bears no relation to the structure of the environment, unlike actionable information, and in particular the AIN. For the purpose of validation, I have used standard tools from multiple-view geometry [53] to reconstruct the trajectory of the vehicle and its relation with the 3D structure of the environment[25] (Fig. 12).

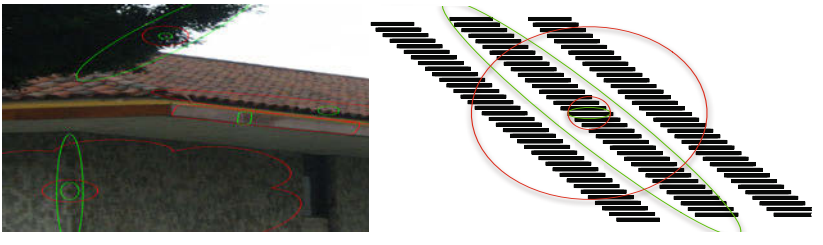


Fig. 5 The same point on an image can be represented, depending on scale, as "structure" (extrema and discontinuities, such as edges and ridges), then "texture" (spatially stationary, or cyclo-stationary, statistics), then again structure (green), and again texture (red) etc. All interpretations must be retained in the representation, rather than selecting one particular scale. One-dimensional signals obey a "causality principle" whereby structure can only be lost, but not created, with increasing scale [51]. This is not the case with two-dimensional images.

6.1 Exploration via Information Pickup

The "ground truth" Entropy Map (Fig. 8 top) and Complete Information Map (Fig. 8 middle) are computed from (real) images collected with a fixed-heading camera with 90^o field-of-view regularly sampled on a 20cm grid and up-sampled/interpolated to a 40×110 mesh (sample views are shown in Fig. 8 overlaid on the map of the room). Complete Information is computed as a sufficient statistic of the light field, that is

[24] Data courtesy of Google, INC.

[25] Courtesy of T. Lee. A poor man's version of this experiment would use Google's pseudo-ground truth for the trajectory, and trust Google Earth to portray images suggestive of the three-dimensional structure of the environment.

Fig. 6 Representational structures: Superpixel tree (top), dimension-two structures (color/texture regions), dimension-one structures (edges, ridges), dimension-zero structures (Harris junctions, Difference-of-Gaussian blobs). Structures are computed at all scales, and a representative subset of (multiple) scales are selected based on the local extrema of their respective detector operators (scale is color-coded in the top figure, red=coarse, blue=fine). Only a fraction of the structures detected are visualized, for clarity purposes. All structures are supported on the Representational Graph, described in the next figure.

as the actionable information of each image computed at each position in space. The traversable space here is restricted to the inside of the room, so the explorer is not allowed to go outside; however, openings due to doors and windows extend the universe to the adjacent rooms and the vegetation outside the window.

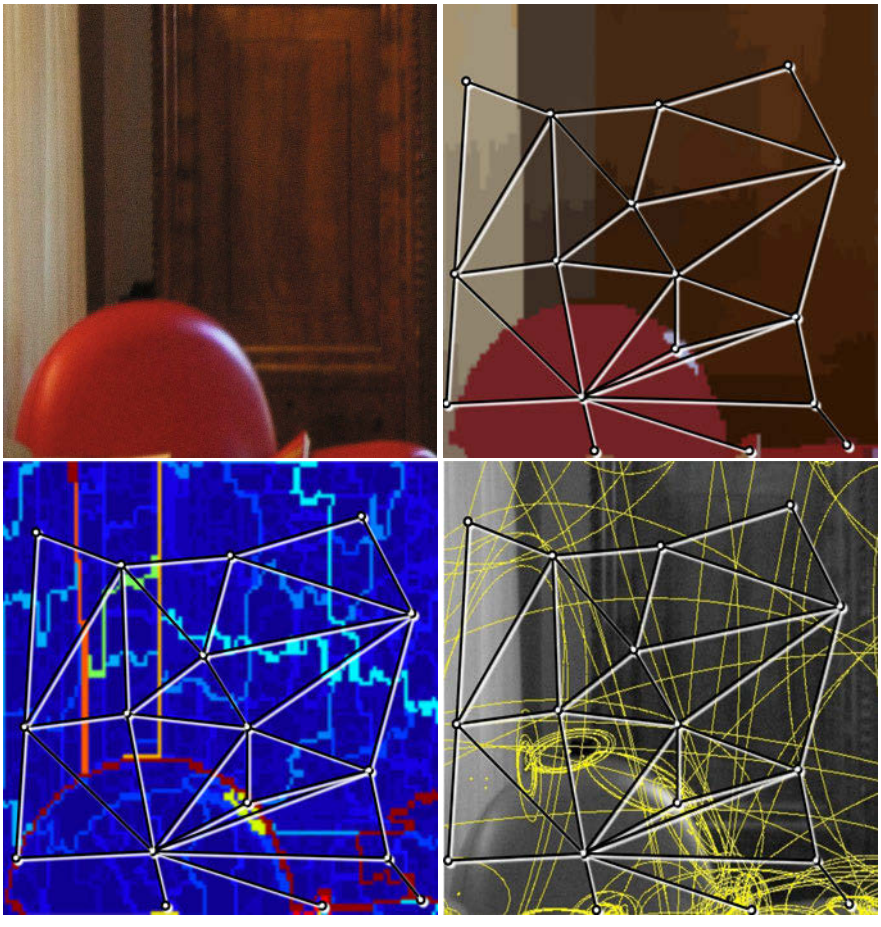


Fig. 7 Representational Graph: (detail, top-left) Texture Adjacency Graph (TAG, top-right); nodes encode (two-dimensional) region statistics (vector-quantized filter-response histograms, or the ART of chromaticity within the region); pairs of nodes, represented by graph edges, encode the likelihood computed by a multi-scale (one-dimensional) edge/ridge detector between two regions; pairs of edges and their closure (graph faces) represent (zero-dimensional) attributed points (junctions, blobs). For visualization purposes, the nodes are located at the centroid of the regions, and as a result the attributed point corresponding to a face may actually lie outside the face as visualized in the figure. This bears no consequence, as geometric information such as the location of point features is discounted in a viewpoint-invariant statistic.

The first agent considered is a **Brownian Explorer**, that follows a random walk governed by the stochastic differential equation (SDE)

$$\begin{cases} dX(t) = V(t)dt; \quad X(0) \sim \mathcal{U}(S \subset \mathbb{R}^2); V(0) = 0 \\ dV(t) = dW(t) \quad \text{a Wiener Process w/ cov. } \sigma^2 \end{cases} \tag{14}$$

Fig. 8 Entropy vs. Actionable Information (first and second from the top) displayed as a function of position for a mobile agent with constant heading and 90^o field-of-view (bright = high; dark = low). Entropy relates to the structure of the image, without regard to the three-dimensional structure of the environment: It is high in the presence of complex textures (wallpaper and wood wainscoting) in the near field as well as complex scenes in the distance. Actionable Information, on the other hand, discounts periodic and stochastic textures, and prefers apertures (doors and windows), as well as specular highlights. Note the region on the right-hand side shows high levels of Actionable Information, proportional to the percentage of the field of view that intercepts the door aperture. Four representative images have been selected, corresponding to a field of view indicated by a colored cone (yellow, green, orange, and blue). Their coding residual is shown below. Note that, except for specular reflections, the complex wallpaper and wood grain does not trigger a high residual, but the opening behind the windows (yellow and blue viewing cone) does. The representational structures computed for every image collected (an approximation of the light field) constitutes the Complete Information, that is not available to the explorer beforehand.

to be integrated in the Îto sense.[26] In practice, we can make do with the discrete-time stochastic process generated by

$$\begin{cases} X(t+dt) = X(t) + V(t)dt; & X(0) \sim \mathcal{U}(S \subset \mathbb{R}^2) \\ V(t+dt) = V(t) + W(t)dt; & W(t) \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2 I) \end{cases} \tag{15}$$

[26] See [46] (p. 6) for a definition and characterization of a Wiener process, and [44] (Chapt. 2 and 5, in particular eq. (2.1) and the rest of Ch. 5.2) for the meaning of the SDE.

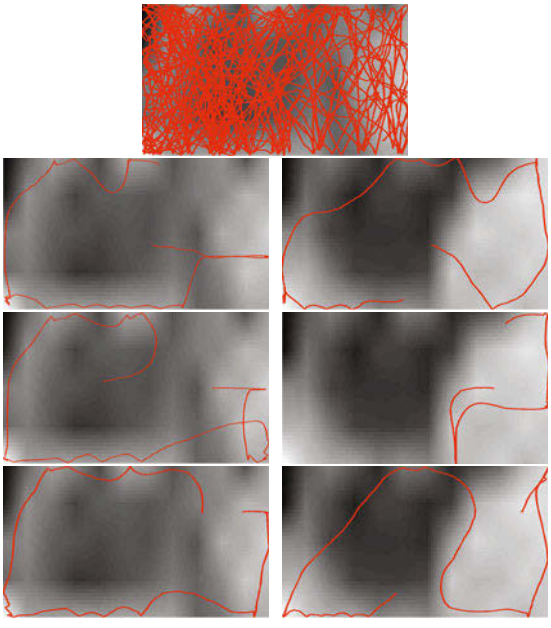


Fig. 9 Brownian (top), Shannonian (left) and Gibsonian (right) Information Pickup. Representative samples of exploration runs are shown. The Shannonian Explorer (left column) is attracted by wallpaper (top edge of each plot) and the foliage outside the window (bottom-left corner of each plot). The Gibsonian Explorer (right column), aims for the window (bottom-left corner of the room) or the door (top-right corner of the room) like a trapped fly, and is similarly repelled by the control law that prohibits escape.

with $V(0) = 0$. The trajectory charted by the Brownian Explorer (e.g., the Roomba vacuum cleaner) is shown in Fig. 9 (top).[27] Clearly, one can do better with vision. For the **Shannonian Explorer** I consider directly the discrete-time model, with a temporal evolution of the entropy $H(I(x,t)|g(t) = X)$ of the image I captured at time t in position X. **The Gibsonian Explorer** seeks to maximize the AIN myopically, as described in the previous section. Representative sample trajectories of the Shannonian and Gibsonian explorations are shown in Fig. 9 (left and right column respectively). The Shannonian Explorer loves wallpaper, complex texture and generally operates regardless of the 3D structure of the scene. The Gibsonian Explorer is claustrophobic: It prefers apertures and attempts to go through windows and doors; the simulation does not allow that, hence it behaves like a fly on glass windows. Note that the Gibsonian explorer is not given the complete information, and can therefore only plan its action based on the AIN. The goal of this

[27] The behavior of our naïf Brownian explorer around the boundaries (Fig. 9) is dictated by a simplified reflection processes: We just invert the component of velocity that causes the crossing of the boundary. A proper simulation would instead use shadow paths following the Reflection Principle as described in [44] (Sect. 2.6.A, p. 79).

experiment is to show that local control strategies, based on image computations, yield space exploration that is compatible with the complete information, that is considered ground truth. In Fig. 10 I experimented with the dilemma between complexity due to near-field texture versus far-away structure, two situations indistinguishable from an image alone. In the ecological approach to perception, however, accommodation is actively controlled, so one can discriminate between complexity due to nearby texture (near-field focus), or to far-away structure (far-field focus).

Fig. 10 Effects of Accommodation: The same scene (top, detail at the bottom) viewed from similar vantage points while focusing in the near (left) and far field (right). Entropy is virtually identical (right is 4% lower, 7.3414nats vs. 7.0451nats), but the complexity on the left is due to the foreground texture, whereas on the right it is due to the structure of the background. Coding length is different, which reflects the self-similarity of the foreground texture (right is 47% higher, 94,375Bytes vs. 138,638Bytes). Actionable Information captures this fact as well (right is 52% higher, 10,939bits vs 16,608bits.) If accommodation is actively controlled, one can easily distinguish nearby texture from far away structure from the feedback signal of the accommodation control.

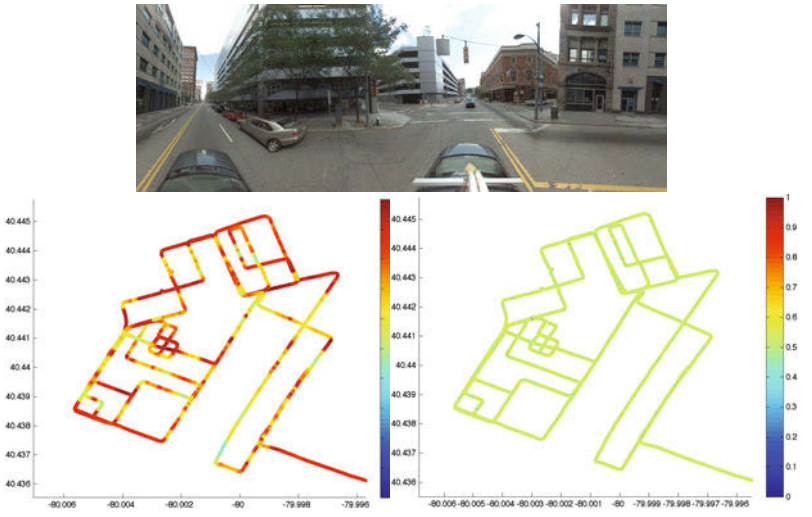


Fig. 11 Google StreetView Dataset. Linear panorama, $2,560 \times 905$ pixels RGB. Entropy (left) and Entropy gradient along the path is shown color-coded at the bottom throughout the 12,000 frame-long sequence. Neither bear any relation to the geometry of the scene.

It is unsurprising, and patent in Fig. 11, that neither Entropy nor its gradient bear any relation to the structure of the scene. In Fig. 12, I show the top view of a 250 frame-long detail with the trajectory and point-wise structure computed from point correspondences using standard tools from multiple-view geometry. For comparison, I also show the pseudo-ground truth provided with the dataset (yellow push-pins). The color-coded trajectory on the bottom shows the entropy gradient, with enhanced color-coding (red is high, blue is low). On the top I show the same for the Actionable Information Gap. It shows peaks at turns and intersections, when large swaths of the scene suddenly become visible. Note that the peaks are both before and after the intersection, as the omni-directional viewing geometry makes the sequence symmetric with respect to forward and backward directions. For the same reason, there is a constant "creation/distruction of data" in the direction of motion due to quantization. The "ground truth" coordinates are rather imprecise, as they would have the vehicle crossing lanes into opposing traffic and into buildings.

Trees, and vegetation in general, attract both the Shannonian and the Gibsonian explorers, as they are photometrically complex, but also geometrically complex because of the fine-scale occlusion structure, visible in the last part of the sequence (right-hand side of the plot; images are shown in Fig. 11). Similar considerations hold for highly specular objects such as cars and glass windows. Although this experiment does not entail active exploration, but only passive motion, it shows that the AIN strongly relates to the structure of the scene, and in particular to its topology.

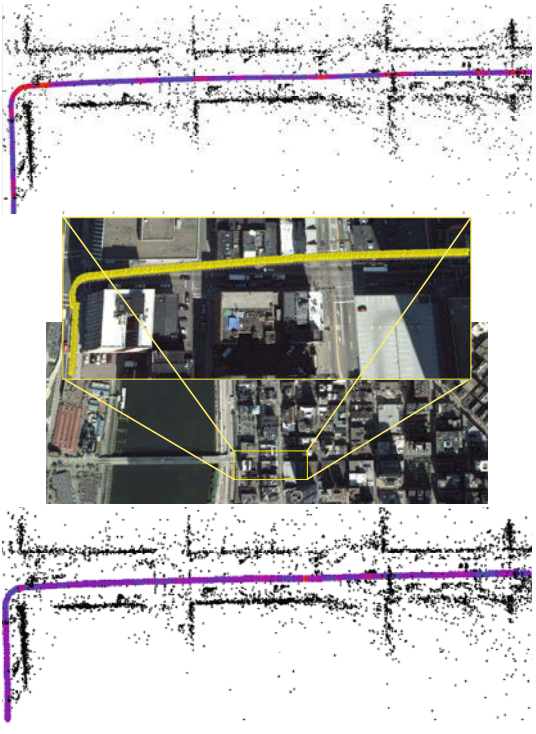


Fig. 12 Navigation via Minimization of the Actionable Information Gap. Actionable Information gap (top) vs. Data Entropy gradient (bottom) color-coded (blue=small, red=large) for a 250-frame long detail of the Google Street View dataset, overlaid with the top-view of the point-wise 3D reconstruction computed using standard multiple-view geometry. For reference, the top-view from Google Earth is shown, together with push-pins corresponding to "ground truth" coordinates. The Entropy gradient (bottom) shows no relation with the 3D structure of the scene. Actionable Information (top), on the other hand, has peaks at turns and intersections, when large portions of the scene become visible (getting into the intersection) and thence disappear (getting out of the intersection).

7 Summary and Discussion

I have presented a characterization of visual information for the purpose of decision and control tasks. Actionable Information is defined as the complexity of the maximal statistic that is invariant to the nuisances. Specifically, I have considered viewpoint and illumination variations, which have been recently shown to admit invariant sufficient statistics. In addition, I have considered occlusion and quantization artifacts, that cannot be inverted, and therefore induce an "information gap" that can be filled by controlling the data acquisition process.

While in traditional Communications Theory "all data matters," in the context of Actionable Information at least a portion of the data is irrelevant, and *the process*

of "extracting information from data" requires a control action. This tie between sensing and control is very prominent in Actionable Information.

I have illustrated these ideas on a simulated exploration task, guided by visual measurements. Whereas a Shannonian Explorer is guided by the complexity of the *data*, a Gibsonian Explorer is guided by the topology of the physical space surrounding it. In both cases, the data consists of images, and no 3D reconstruction, stereo or structure-from-motion is necessary.

This work relates to visual navigation and robotic localization and planning [74, 10, 37]. In particular, [81, 13, 71] propose "information-based" strategies, although by "information" they mean localization and mapping uncertainty based on range data. Range data are not subject to illumination and viewpoint nuisances, which are suppressed by the active sensing, *i.e.* by flooding the space with a known probing signal (*e.g.* laser light or radio waves) and measuring the return. There is a significant literature on vision-based navigation [14, 84, 58, 60, 73, 66, 28, 25, 64, 43], and our experimental section could be characterized simply as occlusion-driven navigation [47, 48, 9]. In most of the literature, stereo or motion are exploited to provide a three-dimensional map of the environment, which is then handed off to a path planner, separating the *photometric* from the *geometric and topological* aspect of the problem. Not only is this separation unnecessary, it is also ill-advised, as the regions that are most informative are precisely those where stereo provides no disparity. Our navigation experiments also relate to Saliency and Visual Attention [38], although there the focus is on navigating the *image*, whereas we are interested in navigating the *scene*, based on image data. In a nutshell, robotic navigation literature is "all scene and no image," the visual attention literature is "all image, and no scene." I bridge the gap by proposing an approach that allows to go "from image to scene, and vice-versa" in the process of Information Pickup. The relationship between visual incentives and spatial exploration has been a subject of interest in psychology for a while [17].

This is not a paper on visual recognition, although it does propose a representation (the Representational Graph) that integrates structures of various dimensions into a unified representation that can, in principle, be exploited for recognition. In this sense, it presents an alternative to [35, 76], that could also be used to compute Actionable Information. However, the rendition of the "primal sketch" [54] in [35] does not guarantee that the construction is "lossless" with respect to any particular task, because there is no underlying task guiding the construction. Our work also relates to the vast literature on segmentation, particularly texture-structure transitions [83]. Alternative approaches to this task could be specified in terms of sparse coding [59] and non-local filtering [15]. I stress the fact that, while no single segmentation is "right" or "wrong," the collection of all possible segmentations, with respect to all possible statistics pooled at all possible scales, is "useful" in the sense of providing pre-computation of the optimization or marginalization functional implicit in any recognition task. This paper also relates to the literature of ocular motion, and in particular saccadic motion. The human eye has non-uniform resolution, which affects motion strategies in ways that are not tailored to engineering systems with uniform

resolution. One could design systems with non-uniform resolution, but mimicking the human visual system is not our goal.

Our work also relates to other attempts to formalize "information" including the so-called Epitome [41], that could be used as an alternative to our Representational Structure if one could compute it fast enough. Furthermore, the Epitome does not capture compactness and locality, that are importantly related to the structure of the scene (occlusions) and its affordances (relationship to the viewer). For instance, if one has same texture patch in different locations in space, these are lumped together, regardless of compactness. Another alternative is the concept of Information Bottleneck, [75], and our approach can be understood as a special case tailored to the statistics and invariance classes of interest, that are task-specific, sensor-specific, and control authority-specific. These ideas can be seen as seeds of a theory of *"Controlled Sensing"* that generalizes Active Vision to different modalities whereby the purpose of the control is to counteract the effect of nuisances. This is different than Active Sensing, that usually entails broadcasting a known or structured probing signal into the environment. Our work also relates to attempts to define a notion of information in statistics [52, 11], economics [55, 4] and in other areas of image analysis [45] and signal processing [33]. Our particular approach to defining the underlying representational structure relates to the work of Guillemin and Golubitsky [32]. Our work also relates to video coding/compression: As I have pointed out, poor man's versions of some of our constructions could be computed using standard operations from the video coding standards. However, I advocated structures that are adapted to the image data (superpixels, TAG, representational graph) rather than on fixed blocks. We can do this because, to achieve invariance to viewpoint, we have no need to encode deformation of these regions, just their correspondence.

Last, but not least, our work relates to Active Vision [1, 12, 8], and to the "value of information" [55, 27, 34, 22, 20]. The specific illustration of the experiment to the sub-literature on next-best-view selection [61, 9]. Although this area was popular in the eighties and nineties, it has so far not yielded usable notions of information that can be transposed to other visual inference problems, such as recognition and 3D reconstruction.

Similarly to previously cited work [81, 13], [26] propose using the decrease of uncertainty as a criterion to select camera parameters, and [3] uses information-theoretic notions to evaluate the "informative content" of laser range measurements depending on their viewpoint.

Clearly, one can raise a number of objections to the concepts defined here, both on mathematical and on philosophical grounds. For start, if we define occlusion as a nuisance, then a sufficient statistic can never be known until we explore the entire world and beyond, for we cannot know what is "on the other side of the hill[28]". However, the sufficient statistics are defined by the task, and if the task is navigation, then a sufficient statistic is aggregated until all openings in a space have been

[28] Occlusions have long fascinated humans both for practical reasons (*e.g.* the Duke of Wellington's quote "All the business of war, and indeed all the business of life, is to endeavor to find out what you don't know by what you do; that's what I called 'guessing what was on the other side of the hill'.") and for aesthetic ones (*e.g.* Leopardi's "L'Infinito").

explored. If the task is recognition of a particular object or class, partial occlusions can be resolved, and total occlusions (*i.e.* the absence of the object of interest in the visual field) requires active search to resolve, and will not end until the object is found. Also, the invariant sufficient statistic described in [72] assume that the image is a Morse function. While Morse functions are dense in $C^2(\mathbb{R}^2 \to \mathbb{R})$, which is dense in $\mathbb{L}^2$, and therefore they can approximate any square-integrable function arbitrarily well, co-dimension one extrema (edges, ridges, valleys) are qualitatively different than elongated blobs. Nevertheless, one could extend the analysis to (multi-scale) edge and ridge detectors, for instance following the guidelines of [50, 18], and still have a thin set that encodes all the actionable information. This extension is the subject of future work.

Acknowledgments. I wish to thank Andrea Vedaldi, Joseph O' Sullivan and Richard Wesel for suggestions leading to a reformulation of Sect. 4.3, Alan Yuille, Serge Belongie and Ying-Nian Wu for discussions, Tahee Lee for processing the Google StreetView dataset, Brian Fulkerson and Jason Meltzer for providing examples, Roger Brockett for feedback and references, and Jitendra Malik for the Tunicate example. This research is supported by FA9550-09-1-0427, FA8650-11-1-7156, and ARO56765-CI. A short version of this paper appears in the Proc. of the Intl. Conf. on Comp. Vision [67]. This material was first presented at the International Computer Vision Summer School (ICVSS) in 2009. I wish to thank Sebastiano Battiato, Roberto Cipolla, and Giovanni Maria Farinella for organizing such a memorable school, and for their kind invitation.

Follow-up work includes the course notes [68], a characterization of detachable object detection, [7, 6], a characterization of the controlled recognition bounds [70], of sufficient exploration [62], and the optimal descriptor TST/BTD [49].

References

1. Aloimonos, J., Weiss, I., Bandyopadhyay, A.: Active vision. International Journal of Computer Vision 1(4), 333–356 (1988)
2. Alvarez, L., Guichard, F., Lions, P.L., Morel, J.M.: Axioms and fundamental equations of image processing. Arch. Rational Mechanics 123 (1993)
3. Arbel, T., Ferrie, F.P.: Informative views and sequential recognition. In: Conference on Computer Vision and Pattern Recognition (1995)
4. Arrow, K.J.: Information and economic behavior. In: Federation of Swedish Industries Stockholm, Sweden (1973)
5. Ayvaci, A., Raptis, M., Soatto, S.: Optical flow and occlusion detection with convex optimization. In: Proc. of Neuro Information Processing Systems (NIPS) (December 2010)
6. Ayvaci, A., Soatto, S.: Efficient model selection for detachable object detection. UCLA Technical Report, March 23 (2011), (submitted)
7. Ayvaci, A., Soatto, S.: Detachable object detection. Technical Report UCLA-CSD-100036 (November 19 February 22, 2011) (2010) (submitted)
8. Bajcsy, R.: Active perception. Proceedings of the IEEE 76(8), 966–1005 (1988)
9. Bajcsy, R., Maver, J.: Occlusions as a guide for planning the next view. IEEE Trans. Pattern Anal. Mach. Intell. 15(5) (May 1993)

10. Batalin, M.A., Sukhatme, G.S.: Efficient exploration without localization. In: Proceedings of IEEE International Conference on Robotics and Automation, 2003. ICRA 2003, vol. 2 (2003)
11. Bernardo, J.M.: Expected information as expected utility. Annals of Stat. 7(3), 686–690 (1979)
12. Blake, A., Yuille, A.: Active vision. MIT Press Cambridge (1993)
13. Bourgault, F., Makarenko, A.A., Williams, S., Grocholsky, B., Durrant-Whyte, H.: Information based adaptive robotic exploration. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), vol. 1 (2002)
14. Brooks, R.: Visual map making for a mobile robot. In: Proceedings of 1985 IEEE International Conference on Robotics and Automation, vol. 2 (1985)
15. Buades, A., Coll, B., Morel, J.M.: A non-local algorithm for image denoising. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2005, vol. 2 (2005)
16. Burns, J.B., Weiss, R.S., Riseman, E.M.: The non-existence of general-case view-invariants. In: Geometric Invariance in Computer Vision (1992)
17. Butler, R.B.: The effect of deprivation of visual incentives on visual exploration motivation in monkeys. Journal of Comparative and Physiological Psychology 50(2), 177 (1957)
18. Candès, E.J., Donoho, D.L.: Curvelets, multiresolution representation, and scaling laws. In: Wavelet Applications in Signal and Image Processing (2000)
19. Caselles, V., Coll, B., Morel, J.-M.: Topographic maps and local contrast changes in natural images. Int. J. Comput. Vision 33(1), 5–27 (1999)
20. Castro, R., Kalish, C., Nowak, R., Qian, R., Rogers, T., Zhu, X.: Human active learning. In: Proc. of NIPS (2008)
21. Chen, H.F., Belhumeur, P.N., Jacobs, D.W.: In search of illumination invariants. In: Proc. IEEE Conf. on Comp. Vision and Pattern Recogn. (2000)
22. Claxton, K., Neumann, P.J., Araki, S., Weinstein, M.C.: Bayesian value-of-information analysis. International Journal of Technology Assessment in Health Care 17(01), 38–55 (2001)
23. Cremers, D., Soatto, S.: Motion competition: a variational approach to piecewise parametric motion segmentation. Intl. J. of Comp. Vision, 249–265 (May 2005)
24. Da Cunha, A.L., Do, M.N., Vetterli, M.: On the information rates of the plenoptic function. In: ICIP, Atlanta, GA (2006)
25. Davison, A.J., Murray, D.W.: Simultaneous localization and map-building using active vision. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7), 865–880 (2002)
26. Denzler, J., Brown, C.M.: Information theoretic sensor data selection for active object recognition and state estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(2), 145–157 (2002)
27. Fogel, E., Huang, Y.F.: Value of Information in System Identification-Bounded Noise Case. Automatica 18(2), 229–238 (1982)
28. Franz, M.O., Schölkopf, B., Mallot, H.A., Bülthoff, H.H.: Learning view graphs for robot navigation. Autonomous Robots 5(1), 111–125 (1998)
29. Gibson, J.J.: The theory of information pickup. In: Contemp. Theory and Research in Visual Perception, p. 662 (1968)
30. Gibson, J.J.: The myths of passive perception. Philosophy and Phenomenological Research 37(2), 234–238 (1976)
31. Gibson, J.J.: The ecological approach to visual perception. LEA (1984)
32. Golubitsky, M., Guillemin, V.: Stable mappings and their singularities. Graduate texts in mathematics, vol. 14 (1974)
33. Good, I.J., Osteyee, D.B.: Information, weight of evidence. The singularity between probability measures and signal detection. Springer, Heidelberg (1974)
34. Gould, J.P.: Risk, stochastic preference, and the value of information. Journal of Economic Theory 8(1), 64–84 (1974)

35. Guo, C., Zhu, S., Wu, Y.N.: Toward a mathematical theory of primal sketch and sketchability. In: Proc. 9th Int. Conf. on Computer Vision (2003)
36. Huang, J., Mumford, D.: Statistics of natural images and models. In: Proc. CVPR, pp. 541–547 (1999)
37. Hughes, S.B., Lewis, M.: Task-driven camera operations for robotic exploration. IEEE Transactions on Systems, Man and Cybernetics, Part A 35(4), 513–522 (2005)
38. Itti, L., Koch, C.: Computational modelling of visual attention. Nature Rev. Neuroscience 2(3), 194–203 (2001)
39. James, K.: On some possible characteristics of information in J. J. Gibson's ecological approach to visual perception. Leonardo 13(2) (1980)
40. Jin, H., Soatto, S., Yezzi, A.: Multi-view stereo reconstruction of dense shape and complex appearance. Intl. J. of Comp. Vis. 63(3), 175–189 (2005)
41. Jojic, N., Frey, B., Kannan, A.: Epitomic analysis of apperance and shape. In: Proc. ICCV (2003)
42. Jones, E., Soatto, S.: Visual-inertial navigation, localization and mapping: A scalable real-time large-scale approach. Intl. J. of Robotics Research (January 17, 2011)
43. Jones, S.D., Andersen, C., Crowley, J.L.: Appearance based processes for visual navigation. In: Processings of the 5th International Symposium on Intelligent Robotic Systems (SIRS 1997), pp. 551–557 (1997)
44. Karatzas, I., Shreve, S.E.: Brownian Motion and Stochastic Calculus. Springer, Heidelberg (1988)
45. Keeler, K.C.: Map representation and optimal encoding for image segmentation. PhD dissertation. Harvard University (October 1990)
46. Kunita, H.: Stochastic differential equations on manifolds. Cambridge University Press (1991)
47. Kutulakos, K.N., Dyer, C.R.: Global surface reconstruction by purposive control of observer motion. Artificial Intelligence 78(1-2), 147–177 (1995)
48. Kutulakos, K.N., Jagersand, M.: Exploring objects by invariant-based tangential viewpoint control. In: Proceedings of International Symposium on Computer Vision, pp. 503–508 (1995)
49. Lee, T., Soatto, S.: Learning and matching multiscale template descriptors for real-time detection, localization and tracking. In: Proc. IEEE Conf. on Comp. Vision and Pattern Recogn., pp. 1457–1464 (2011)
50. Lindeberg, T.: Edge Detection and Ridge Detection with Automatic Scale Selection, vol. 30. Cambridge University Press (1998)
51. Lindeberg, T.: Principles for automatic scale selection. Technical report, KTH, Stockholm, CVAP (1998)
52. Lindley, D.V.: On a measure of the information provided by an experiment. Annals of Math. Stat. 27(4), 986–1005 (1956)
53. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.: An invitation to 3D vision, from images to geometric models. Springer, Heidelberg (2003)
54. Marr, D.: Vision. W.H.Freeman & Co. (1982)
55. Marschak, J.: Remarks on the economics of information. Contributions to Scientific Research in Management (1960)
56. 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)
57. Mumford, D., Gidas, B.: Stochastic models for generic images. Quarterly of Applied Mathematics 54(1), 85–111 (2001)
58. Newman, P., Ho, K.: SLAM-loop closing with visually salient features. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation. ICRA 2005, pp. 635–642 (2005)
59. Olshausen, B.A., Field, D.J.: Sparse coding with an overcomplete basis set: A strategy employed by V1? In: Vision Research (1998)

60. Peruch, P., Vercher, J.-L., Gauthier, G.M.: Acquisition of spatial knowledge through visual exploration of simulated environments. Ecological Psychology 7(1), 1–20 (1995)
61. Pito, R., Co, I.T., Boston, M.A.: A solution to the next best view problem for automated surface acquisition. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(10), 1016–1030 (1999)
62. Pretto, A., Chiuso, A., Soatto, S.: Sufficient exploration for navigation and recognition. UCLA Technical Report (January 17, 2011) (submitted)
63. Robert, C.P.: The Bayesian Choice. Springer, New York (2001)
64. Se, S., Lowe, D., Little, J.: Vision-based mobile robot localization and mapping using scale-invariant features. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA (2001)
65. Shao, J.: Mathematical Statistics. Springer, Heidelberg (1998)
66. Sim, R., Dudek, G.: Effective exploration strategies for the construction of visual maps. In: Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), vol. 4 (2003)
67. Soatto, S.: Actionable information in vision. In: Proc. of the Intl. Conf. on Comp. Vision (October 2009)
68. Soatto, S.: Steps Toward a Theory of Visual Information. Technical Report UCLA-CSD100028 (September 13, 2010)
69. Soatto, S.: Texture, structure and visual matching. Technical Report UCLA-CSD-100033 (October 11, 2010)
70. Soatto, S., Chiuso, A.: Controlled recognition bounds for scaling and occlusion channels. In: Proc. of the Data Compression Conference (March 2011)
71. Stachniss, C., Grisetti, G., Burgard, W.: Information gain-based exploration using rao-blackwellized particle filters. In: Proc. of RSS (2005)
72. Sundaramoorthi, G., Petersen, P., Soatto, S.: On the set of images modulo viewpoint and contrast changes. Technical Report UCLA-CSD090005 (2009) (submitted)
73. Taylor, C.J., Kriegman, D.J.: Vision-based motion planning and exploration algorithms for mobile robots. IEEE Trans. on Robotics and Automation 14(3), 417–426 (1998)
74. Thrun, S.: Learning metric-topological maps for indoor mobile robot navigation. Artificial Intelligence 99(1), 21–71 (1998)
75. Tishby, N., Pereira, F.C., Bialek, W.: The information bottleneck method. In: Proc. of the Allerton Conf. (2000)
76. Todorovic, S., Ahuja, N.: Extracting subimages of an unknown category from a set of images. In: Proc. IEEE Conf. on Comp. Vis. and Patt. Recog. (2006)
77. Turing, A.M.: The chemical basis of morphogenesis. Phil. Trans. of the Royal Society of London, ser. B (1952)
78. Vedaldi, A., Soatto, S.: Features for recognition: viewpoint invariance for non-planar scenes. In: Proc. of the Intl. Conf. of Comp. Vision, pp. 1474–1481 (October 2005)
79. Vedaldi, A., Soatto, S.: Quick Shift and Kernel Methods for Mode Seeking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 705–718. Springer, Heidelberg (2008)
80. Wertheimer, M.: Laws of organization in perceptual forms. In: Ellis, W.D. (ed.) A Sourcebook of Gestalt Psychology, pp. 331–363. Harcourt, Brace and Company (1939)
81. Whaite, P., Ferrie, F.P.: From uncertainty to visual exploration. IEEE Transactions on Pattern Analysis and Machine Intelligence 13(10), 1038–1049 (1991)
82. Wiener, N.: Cybernetics, or Control and Communication in Men and Machines. MIT Press (1949)
83. Wu, Y.N., Guo, C., Zhu, S.C.: From information scaling of natural images to regimes of statistical models. Quarterly of Applied Mathematics 66, 81–122 (2008)
84. Zhang, H., Ostrowski, J.P.: Visual motion planning for mobile robots. IEEE Transactions on Robotics and Automation 18(2), 199–208 (2002)

Learning Binary Hash Codes for Large-Scale Image Search

Kristen Grauman and Rob Fergus

Abstract. Algorithms to rapidly search massive image or video collections are critical for many vision applications, including visual search, content-based retrieval, and non-parametric models for object recognition. Recent work shows that *learned binary projections* are a powerful way to index large collections according to their content. The basic idea is to formulate the projections so as to approximately preserve a given similarity function of interest. Having done so, one can then search the data efficiently using hash tables, or by exploring the Hamming ball volume around a novel query. Both enable sub-linear time retrieval with respect to the database size. Further, depending on the design of the projections, in some cases it is possible to bound the number of database examples that must be searched in order to achieve a given level of accuracy.

This chapter overviews data structures for fast search with binary codes, and then describes several supervised and unsupervised strategies for generating the codes. In particular, we review supervised methods that integrate metric learning, boosting, and neural networks into the hash key construction, and unsupervised methods based on spectral analysis or kernelized random projections that compute affinity-preserving binary codes. Whether learning from explicit semantic supervision or exploiting the structure among unlabeled data, these methods make scalable retrieval possible for a variety of robust visual similarity measures. We focus on defining the algorithms, and illustrate the main points with results using millions of images.

Kristen Grauman
Dept. of Computer Science, University of Texas, Austin, USA
e-mail: grauman@cs.utexas.edu

Rob Fergus
Courant Institute, New York University, USA
e-mail: fergus@cs.nyu.edu

R. Cipolla et al. (Eds.): Machine Learning for Computer Vision, SCI 411, pp. 49–87.
springerlink.com

1 Introduction

Huge collections of images and video are increasingly available, arising in domains as diverse as community photo collections, scientific image data of varying modalities, news media, consumer catalogs, or surveillance archives. In the last decade in particular, user-generated content is widely stored and shared on the Web. Numbering in to the tens or hundreds of billions, their sheer size poses a challenge for conventional computer vision techniques. For example, every *minute* more than 20 hours of new video are uploaded to YouTube and 100,000 photos are uploaded to Facebook. Clearly, even real-time methods would be incapable of coping with this deluge of data. Consequently, researchers are exploring new representations and approaches to search large-scale datasets.

At the core of visual search is the nearest neighbor problem: given a query, which items in the database are most like it? Despite the simplicity of this problem statement, fast and accurate nearest neighbor search can enable a spectrum of important applications (see Figure 1).

Efficient algorithms to address the basic similarity search task have received much attention over the years, yielding a variety of tree-based and hashing-based algorithms [22, 9, 66, 44, 47]. However, while applicable to visual data in certain cases, traditional methods often fall short of technical demands inherent to our setting:

- **High-dimensional data.** First of all, good descriptors for images or videos typically live in a high-dimensional space, easily numbering thousands or more dimensions. At the same time, the volume of data quickly strains memory, and disk access is slow. Both aspects argue for mapping to a more compact representation, and/or developing approximate search methods whose error and storage requirements do not blow up with the input dimensionality.

- **Structured input spaces and specialized similarity functions.** Secondly, visual data need not fit neatly within a vector space representation at all. More sophisticated descriptors built on graphs, sets, or trees are often appealing, as they more more closely model the real structure at hand. For example, an image might be described by a planar graph over its component regions, or a video clip may be encoded as a set of loosely ordered keyframes. Alongside such structured representations, researchers have developed specialized affinity or *kernel* functions to accommodate them that are accurate, but would be prohibitively costly to apply naively in the search setting. Thus, there is a clear need for flexibility in the similarity measures supported.

- **Availability of external supervision.** Finally, the complex relationships intrinsic to visual data can be difficult to capture with manually-defined features and metrics. There is a well-known gap between the low-level cues one might pull from an image or video, and the high-level semantics one would like preserved in a content-based search. Access to external knowledge—in the form of labeled

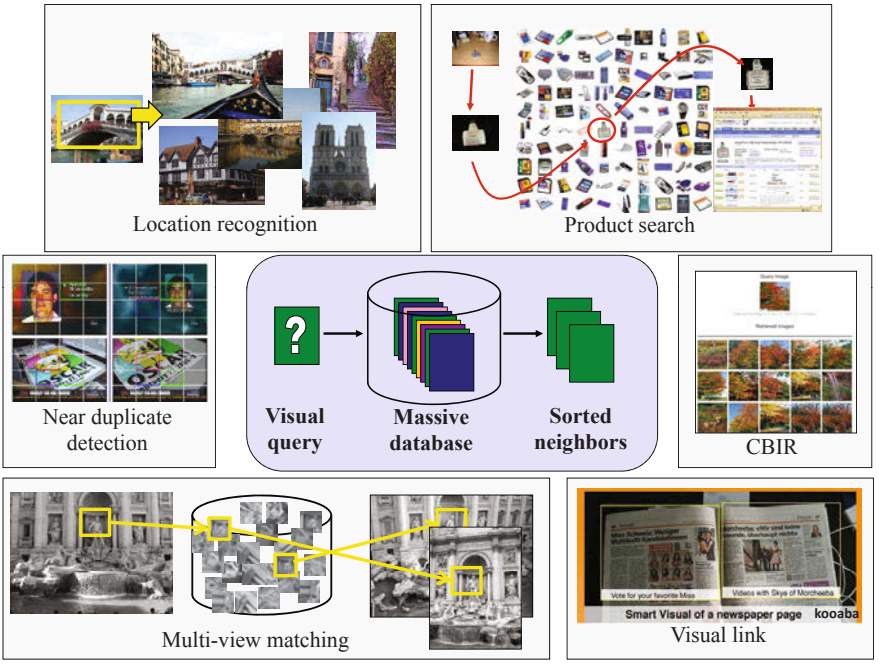


Fig. 1 Large-scale visual search underlies many interesting applications. *Figure credits: Product search image is courtesy of Yeh et al. [75], CBIR image is courtesy of Iqbal and Aggarwal [35], near duplicate detection image is courtesy of Xu et al. [74], and visual link image is from kooaba.com.*

instances or target similarity constraints—would ideally inform the comparisons. This suggests learning should somehow be integrated into the indexing approach or representation.

The central theme of this chapter is the construction and use of binary representations for visual search, as a means to address the above requirements. The main idea is to compute binary projections such that a given similarity function of interest is approximately preserved in Hamming space. Having done so, one can then search the data efficiently using hash tables, or by exploring the Hamming ball volume around a novel query. Depending on the design of the projections, guarantees on the relative retrieval cost and error are sometimes possible.

Why do we explicitly target binary representations? Not only do they fit well with hashing and Hamming-ball search strategies, but also we will see that by carefully maximizing the information carried by each bit, we can achieve far more compact representations for a given storage budget than using real-valued descriptors (which effectively use groups of bits, i.e., 32 for a single-precision real). Clearly, when the datasets are $O(10^{11})$ in size, minimizing the storage overhead is vital, as discussed above.

The algorithms for learned binary projections that we discuss have important ties and contrasts to previous work in dimensionality reduction and distance learning. Dimensionality reduction is a classic problem in machine learning, but the variety of existing techniques are not necessarily well-suited to this domain. Many approaches (such as Isomap [64], LLE [55], or t-SNE [67]) do not scale to large datasets since their complexity is quadratic (or worse) in the number of data points. Furthermore, with large datasets, the ability to project new instances is vital, since it allows parallelization and asynchronous processing of the data. Yet many classic approaches [64, 55, 67] lack an explicit mapping function to the low-dimensional space, making "out-of-sample" projections problematic. In contrast, the techniques we describe have an explicit mapping to the low-dimensional space, allowing the binary representation to be quickly computed.

Effective retrieval requires a representation that places images of semantically similar content close and dissimilar content far apart. Traditional approaches from content-based image retrieval (CBIR) rely on simple representations—for example, using color or edge histograms (see [18] for a review of such methods). However, significant improvements over these representations have proven difficult to achieve through hand-crafting alone. Recent efforts have therefore focused on *learning* good representations using labeled training data (see text below for references). In practice, this can be viewed as a supervised form of dimensionality reduction or distance learning. The work we describe fits into this area, but with a focus on deriving techniques that produce binary embeddings instead of real-valued ones.

In the following section, we first briefly review primary data structures for fast search with binary codes. Having established the search protocols, we then devote the majority of the text to explaining several strategies to generate binary codes. We organize them roughly around the degree of supervision assumed: in Section 3 we describe supervised methods that integrate metric learning, boosting, or neural networks into the hash key construction, and in Section 4 we describe unsupervised methods that use spectral analysis or kernelized random projections to compute affinity-preserving binary codes. While the former exploit explicit supervision from annotated training data, the latter exploit the structure among a sample of unlabeled data to learn appropriate embeddings. We include some example empirical results to illustrate key points.

We refer the reader to the original publications that introduced the algorithms for more details, particularly [37, 40, 57, 60, 72, 39]. While we present the methods in the context of visual retrieval tasks, in fact the core methods are not specific to images in any way, and could be applied to search other forms of data as well.

2 Search Algorithms for Binary Codes

The majority of this chapter is devoted to the construction of compact binary codes, either using label information (Section 3) or just by mimicking the neighborhood structure of the original input space (Section 4). However, mindful of our overall

Dataset	LabelMe	Web
# datapoints	2×10^4	1.29×10^7
Gist vector dim.	512	384
Method	**Time (s)**	**Time (s)**
Spill tree - Gist vector	1.05	-
Brute force - Gist vector	0.38	-
Brute force - 30 bit binary	4.3×10^{-4}	0.146
″ - 30 bit binary, M/T	2.7×10^{-4}	0.074
Brute force - 256 bit binary	1.4×10^{-3}	0.75
″ - 256 bit binary, M/T	4.7×10^{-4}	0.23
Sem. Hashing - 30 bit binary	6×10^{-6}	6×10^{-6}

Fig. 2 Performance of brute-force exhaustive search and Semantic Hashing [56] on two datasets: LabelMe (20,000 images, 512 dimensional Gist input) and Tiny Images (12.9 million, 384 dimensions). The timings for a single query are shown, using a range of different approaches. A spill tree (*kd*-tree variant) [44] is worse than linear search due to the high dimensionality of the input. Reducing the Gist descriptors to binary codes (30-bit and 256-bits) makes linear search very fast ($<$ 1ms/query) on small datasets, but only moderately quick for the larger Tiny Images ($<$ 1s/query). Linear search is easily parallelized (e.g., by using multiple CPU cores—see entries labeled M/T). Semantic hashing is extremely quick ($< 1\mu$s), regardless of dataset size, but can only be applied to compact codes. In contrast, the query time for linear search simply scales linearly with the code length (and database size).

goal, we now explain how these binary codes can be used to perform the nearest-neighbor search.

In general for large-scale retrieval, the most important property is that the search time complexity be sub-linear in the number of database examples. Additionally, given the distributed nature of large-scale computing, the ability to parallelize the search is important for practical applications. In the particular context of binary codes, as we consider here, retrieval involves finding all examples that have a zero or small Hamming distance from the query, where the Hamming distance between two binary vectors is the number of bits that differ between them.

To satisfy these requirements, we consider variants of *hashing*. Hashing is quick and has minimal storage requirements beyond the binary data vectors themselves. It uses all dimensions of the binary codes (bits) in parallel to perform retrieval. This is in contrast to tree-based algorithms such as *kd*-trees, where each example is found by making a series of binary decision to traverse the tree, each decision (bit) being conditional on the choices above.

2.1 Linear Scan in Hamming Space

The most straightforward solution is a brute-force linear scan—that is, to compute the Hamming distance between the query vector and every vector in the database. Although this scales linearly with the size of the database, the constant is typically

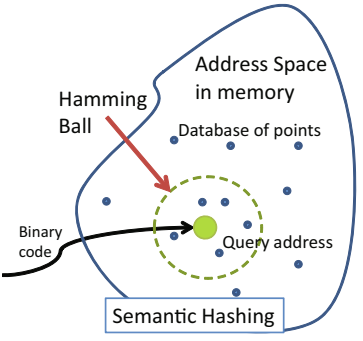


Fig. 3 Semantic Hashing [57] treats the query vector as an address in memory. A Hamming ball is explored by perturbing bits within the query. Nearby points will thus be found by direct memory look-up.

very small, and thus it can be practical for surprisingly large datasets. It is also worth noting that the memory overhead for this approach is virtually nil.

Computing the Hamming distance between two vectors requires two steps: (i) compute the XOR and (ii) count the number of 1's in the resulting vector. Both operations can be performed extremely quickly on modern CPUs, with parallel XOR operations being part of the SSE instruction set. The parallelism can also be trivially extended at the machine level, with each server in a cluster searching a different subset of the database.

In Figure 2 we show timings for brute-force evaluation using 30- and 256-bit codes on two datasets, one of 20,000 vectors, the other 12.7 million. For the 30-bit codes, a 2.0Ghz CPU is able to compare 50 million pairs per second, while for 256-bit codes this increases to 120 million per second. Thus, the brute-force approach scales gracefully to longer code lengths, as we would expect. Figure 2 also shows the use of two cores instead of one nearly doubles the speed, confirming that it is easy parallelized. These results demonstrate that it is a viable approach for moderately large datasets.

2.2 Semantic Hashing

Salakhutdinov and Hinton proposed a nearest-neighbors technique for binary vectors called *Semantic Hashing* whose speed is *independent of the number of data points* [57]. Each binary vector corresponds to an address in memory. Matches to a query vector are found by taking the query vector and systematically perturbing bits within it, so exploring a Hamming ball around the original vector. Any neighbors in the database that fall within this ball will be returned as neighbors. See Figure 3 for an illustration.

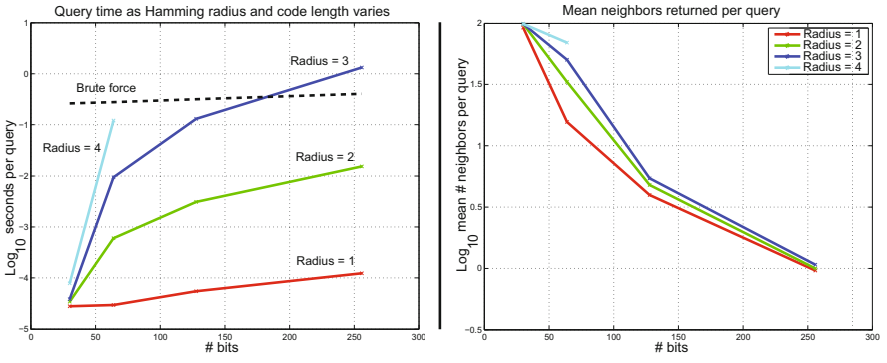


Fig. 4 Left: Query time as a function of code length, for different radii of the Hamming ball. Right: Mean number of returned neighbors as a function of #bits and radius of Hamming ball. Beyond 64 bits or so, the method is no longer significantly faster than brute force methods, and there is a risk that many queries will not find any neighbors. Increasing the radius slows the method down significantly, but does not significantly improve the number of neighbors found.

The approach has two main advantages: (i) provided the radius of the Hamming ball is small, it is extremely quick (i.e., μs, see Figure 2, bottom row); and (ii) constructing the database is also very fast, e.g., compared to *kd*-tree type data structures.

However, a major drawback is that it breaks down for long code vectors, since the mean Hamming distance between points becomes large and the volume of the Hamming ball (which is `nchoosek`(dimension,radius)) becomes prohibitive to explore. To be concrete, suppose we have a code length of 100 bits. The mean distance to a query's closest neighbor may well be quite large, e.g., differing in 7 bits or more. However, if one can afford only a Hamming ball radius search of 3, then very often a query will not find any neighbors within that restricted search volume (see Figure 4, right). Another problem is that Semantic Hashing requires a contiguous block of memory, which becomes impractical for vectors beyond $l = 32$ bits (corresponding to a 4Gb block).[1]

This practical restriction motivates using sophisticated embedding methods to preserve information within a very compact code. For example, neural network-based methods that we will describe in Section 3.3.2 can be used to train a 30-bit descriptor that yields μs retrieval times with Semantic Hashing, as shown in Figure 2 (bottom row). Alternatively, one can employ hashing algorithms that are designed to map similar instances to the *same* bin, thus avoiding the need for Hamming ball exploration or contiguous memory. We discuss such an approach next.

[1] This can be circumvented by introducing a second randomized hash function that maps the 2^l block of memory down to the permissible memory budget. An appropriate randomized hash function will introduce few collisions and be quick, and thus will not impact query accuracy or speed.

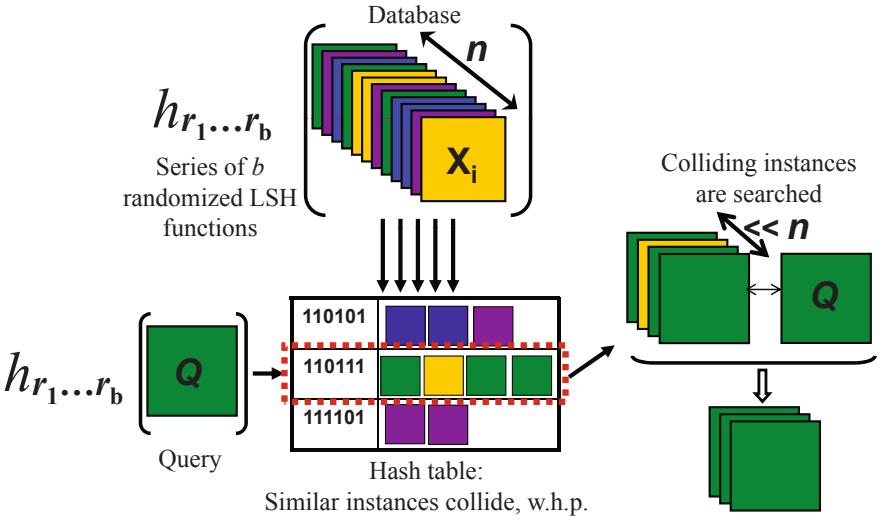


Fig. 5 Locality Sensitive Hashing (LSH) uses hash keys constructed so as to guarantee collision is more likely for more similar examples [33, 23]. Once all database items have been hashed into the table(s), the same randomized functions are applied to novel queries. One exhaustively searches only those examples with which the query collides.

2.3 *Locality Sensitive Hashing*

While the Semantic Hashing approach is simple and extremely effective for very compact codes, it has the practical limitations discussed above, and also lacks formal guarantees on the search quality obtained. *Locality Sensitive Hashing* is a randomized hashing framework introduced earlier by Gionis, Indyk, and Motwani that counters some of these shortcomings, and allows a user to explicitly control the similarity search accuracy and search time tradeoff [23].

The main idea in Locality Sensitive Hashing (LSH) is to insert database items into a hash table such that similar things fall in the same bucket, with high probability [33, 23, 1, 12]. Intuitively, if only highly similar examples collide in the hash table (i.e., are assigned the same *hash key*), then at query time, directly hashing to a stored bucket will reveal the most similar examples, and only those need to be searched. See Figure 5 for a visualization of the method. The hash keys generally consist of low-dimensional binary strings; each database item is mapped to b bits, where each bit is generated independently by a valid locality-sensitive hash function.

Assuming such projections can be appropriately formed, there are algorithms to retrieve the approximate neighbors for a query using hash tables or related data structures [23, 1, 12]. The neighbors are "approximate" in that they are within some ε error of the true near neighbor, and the bounds for the search time are tied to this approximation error. For example, the query time for retrieving $(1+\varepsilon)$-near neigh-

bors can be bounded by $O(n^{1/(1+\varepsilon)})$ for the Hamming distance using appropriate hash functions [23]. This allows a trade-off between the sub-linear retrieval time achieved and the extent to which the returns mimic an exhaustive linear scan, even for high-dimensional input data. Thus, the hashing strategy has important advantages over tree-based techniques that are known to perform poorly in practice for high-dimensional data (e.g., *kd*-trees). Further, because LSH aims to have all relevant instances collide in the same bucket (hash code), its query-time cost depends only linearly on the number of bits used, and does not require a scan of neighboring hash codes (in contrast to Semantic Hashing above).

Note that while early definitions of LSH functions were designed to preserve a given geometric distance, work since has explored ways to formulate *learned* locality-sensitive functions amenable to a target task, or functions that are sensitive to a family of kernel functions, as we will see later in the chapter in Sections 3.2 and 4.3.

More formally, suppose we have a database consisting of data points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$. Given an input query $\boldsymbol{q}$, we are interested in finding those items in the database that are most similar to the query, under some defined measure of similarity or distance (which we will discuss in more detail below). The hash keys are constructed by applying b binary-valued hash functions $h_1,\ldots,h_b$ to each of the database objects, where each h_i is a random sampling from an LSH function family $\mathcal{H}$.

Nearest Neighbor-based LSH Definition. One formulation of LSH [12] describes valid locality-sensitive functions by equating collision probabilities with a similarity score; that is, each hash function $h_{\mathcal{H}}$ drawn from the distribution $\mathcal{H}$ must satisfy:

$$p[h_{\mathcal{H}}(\boldsymbol{x}_i) = h_{\mathcal{H}}(\boldsymbol{x}_j)] = \text{sim}(\boldsymbol{x}_i,\boldsymbol{x}_j), \tag{1}$$

where $\text{sim}(\boldsymbol{x}_i,\boldsymbol{x}_j) \in [0,1]$ is the similarity function of interest.

The preprocessing of the database items is as follows. After computing the projections for all n database inputs, one then forms $M = 2n^{1/(1+\varepsilon)}$ random permutations of the bits. If we think of the database hash keys as an $n \times b$ matrix, that means we randomly permute the vector $[1,2,\ldots,b]$ M times, and use each permutation as indices to reorder the columns of the hash key matrix. Then each list of permuted hash keys is sorted lexicographically to form M "sorted orders". Given a novel query, its hash key indexes into each sorted order with a binary search, and the $2M$ nearest examples found contain the approximate nearest neighbors. This procedure requires searching $O(n^{1/(1+\varepsilon)})$ examples using the original distance function of interest to obtain the $k = 1$ approximate nearest neighbor (NN). See [12] for more details. We will return to this definition below when discussing forms of supervised and unsupervised LSH function generation.

Radius-based LSH Definition. While the above provides guarantees for approximating nearest neighbor search for a similarity function, another related formulation of LSH provides guarantees in terms of the likelihood of collision with a query's r-radius neighbors (i.e., where the goal is to retrieve a database item within a given radius of the query). Let $d(\cdot,\cdot)$ be a distance function over items from a set S, and for

any item $\boldsymbol{p} \in S$, let $B(\boldsymbol{p}, r)$ denote the set of examples from S within radius r from $\boldsymbol{p}$. Let $h_{\mathcal{H}}$ denote a random choice of a hash function from the family $\mathcal{H}$. The family $\mathcal{H}$ is called $(r, r(1+\varepsilon), p_1, p_2)-$sensitive [23] for $d(\cdot,\cdot)$ when, for any $\boldsymbol{q}, \boldsymbol{p} \in S$,

- if $\boldsymbol{p} \in B(\boldsymbol{q}, r)$ then $p[h_{\mathcal{H}}(\boldsymbol{q}) = h_{\mathcal{H}}(\boldsymbol{p})] \geq p_1$,
- if $\boldsymbol{p} \notin B(\boldsymbol{q}, r(1+\varepsilon))$ then $p[h_{\mathcal{H}}(\boldsymbol{q}) = h_{\mathcal{H}}(\boldsymbol{p})] \leq p_2$.

For a family of functions to be useful, it must satisfy $p_1 > p_2$. Note that the probability of collision for close points is thus at least p_1^k, while for dissimilar points it is at most p_2^k.

During a preprocessing stage, all database points are mapped to a series of l hash tables indexed by independently constructed $g_1, \ldots, g_l$, where each g_i is a b-bit function. Then, given a query $\boldsymbol{q}$, an exhaustive search is carried out only on those examples in the union of the l buckets to which $\boldsymbol{q}$ hashes. These candidates contain the (r, ε)-nearest neighbors (NN) for $\boldsymbol{q}$, meaning if $\boldsymbol{q}$ has a neighbor within radius r, then with high probability some example within radius $r(1+\varepsilon)$ is found.[2] More recent work includes an LSH formulation for data in Euclidean space, with improved query time and data structures [1].

Additional notes on LSH. Intuitively the concatenation of b bits into hash keys decreases the false positive rate (we are more selective in what things will collide), whereas the aggregation of search candidates from l independently generated tables increases the recall (we are considering more randomized instances of the functions).

Early work by researchers in the theory community designated LSH function families for Hamming distance, ℓ_p norms, and the inner product [17, 12], as well as embedding functions to map certain metrics into Hamming space (e.g., the Earth Mover's Distance [34]). Given the attractive guarantees of LSH and the relatively simple implementation, vision and machine learning researchers have also explored novel hash function families so as to accommodate fast retrieval for additional metrics of interest. In particular, in this chapter we highlight hash functions for learned Mahalanobis metrics (Section 3.2) and kernel functions (Section 4.3).

2.4 Recap of Search Strategy Tradeoffs

Whereas the Semantic Hashing technique discussed above essentially takes an *embedding* strategy, where similarity ought to fall off smoothly as one looks at more distant codes in Hamming space, Locality Sensitive Hashing takes a direct *hashing* strategy, where similar items ought to map to the same hash key (i.e., Hamming distance = 0). LSH does not entail the bit-length restriction of Semantic Hashing. Memory usage with LSH is typically greater, however, assuming one opts to

[2] For example, in [23] an LSH scheme using projections onto single coordinates is shown to be locality-sensitive for the Hamming distance over vectors. For that hash function, $\rho = \frac{\log p_1}{\log p_2} \leq \frac{1}{1+\varepsilon}$, and using $l = n^{\rho}$ hash tables, a $(1+\varepsilon)$-approximate solution can be retrieved in time $O(n^{\frac{1}{(1+\varepsilon)}})$.

mitigate the 0-threshold Hamming distance by expanding the search to multiple independently generated hash tables.[3] Furthermore, whereas a user of Semantic Hashing specifies a radius of interest in the embedded Hamming space, a user of LSH (for the radius-based search variant) specifies the radius of interest in the original feature space.

Perhaps the main distinction between the methods, however, is the degree to which the search and embedding procedures are integrated. In Semantic Hashing, the procedure to construct the binary embedding is performed independently (without knowledge of) the ultimate search data structure, whereas in LSH, the binary projections and search structure are always intertwined. This distinction can be viewed as a pro or con for either hashing implementation. The elegance and bounds of LSH are a potential advantage, but the restriction of designating appropriate LSH functions limits its flexibility. On the other hand, Semantic Hashing has the flexibility of choosing various learning algorithms to form the binary projections, but its behavior is less predictable with respect to an exhaustive linear scan.

While we will discuss the hash code construction techniques below in the context of one hashing implementation or the other, in practice a user could incorporate either one (or the linear scan), simply keeping the above tradeoffs in mind.

3 Supervised Methods for Learning Binary Projections

The quality of retrieval results clearly will depend on the chosen image representation as well as the distance metric used to compare examples. Ideally, these two components would together accurately reflect the instances' true relationships, such that relevant database items have a small distance to the query, and irrelevant items have a large distance. While a generic distance function (such as an L_p norm) may be more manageable computationally for large-scale search, it may or may not nicely match the desired relationships for a given application. Instead, if we have access to some form of supervision on a subset of examples, then we can attempt to *learn* how to compare them. General supervised classification methods as well as advances in metric learning over the last several years make it possible to fine-tune parametric distance functions [73, 5, 26, 59, 30, 71, 24, 19, 31, 16, 4].

Furthermore, we can attempt to simultaneously learn binary projections that reflect those specialized comparisons, thereby enabling fast Hamming space comparisons. Addressing both aspects generally entails optimizing the metric parameters according to data labeled by their classes or known distance relationships, while also balancing a preference for compact projections.

In this section, we describe two such approaches in detail. The first approach generates randomized hash functions that are locality-sensitive for learned Mahalanobis metrics, exploiting a sparse set of similarity constraints on tuples of points

[3] In practice, a common implementation hack is to simply look at nearby bins according to Hamming distance, similar to Semantic Hashing, even if not necessarily using addresses as the bin index.

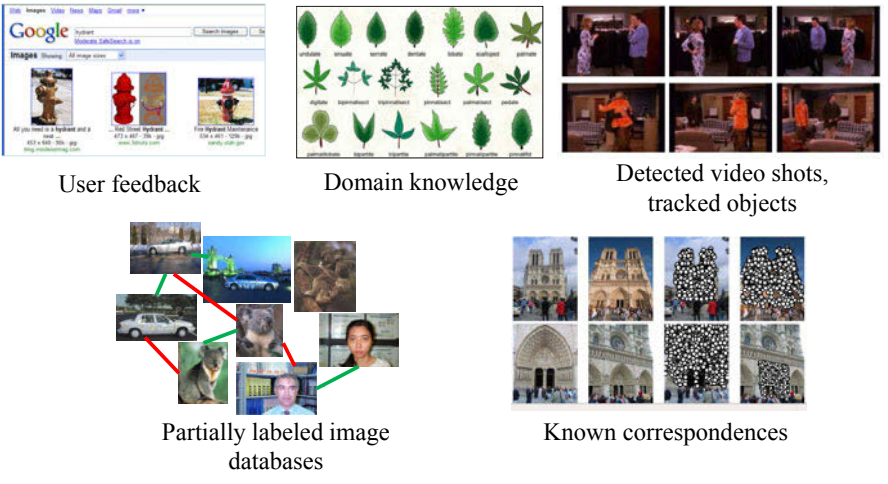


Fig. 6 Supervision about which instances should be near or far from one another could come from a variety of sources.

(Section 3.2). The second approach learns a Hamming embedding from a set of labeled training images (Section 3.3). Both map similar examples to be close-by in a binary space, while keeping dissimilar examples far apart. They differ primarily in the types of learning algorithms integrated to preserve those constraints, and secondarily in the hashing implementation employed alongside.

3.1 Forms of Supervision to Define Semantic Similarity

In the various supervised methods for code construction we discuss, the external supervision can take a variety of forms: labels or feedback on instances can specify those which ought to cluster together, relative judgments on triples of instances can specify their ideal relationships, or the desired nearest neighbor lists for a sample of points can specify those that need to remain close. As such, the techniques are suitable for enhancing nearest neighbor categorization as well as similarity search for content-based retrieval. Figure 6 highlights some possible sources of similarity constraints in the visual search domain.

When similarity constraints are non-exhaustive across a set of training data, we will represent them as sets of pairs: a set $\mathcal{S}$ containing pairs that should remain close (similar), and a set $\mathcal{D}$ containing pairs that should remain far (dissimilar). Alternatively, if we have complete pairwise information on all N training instances, we can think of the semantic information stored in an $N \times N$ matrix. To represent binary similarity constraints, the i, j-th entry in that matrix is 1 if two instances are meant to be close, 0 if far (e.g., with a discrete set of class labels, the entry is 1 for any same-class pairs of points).

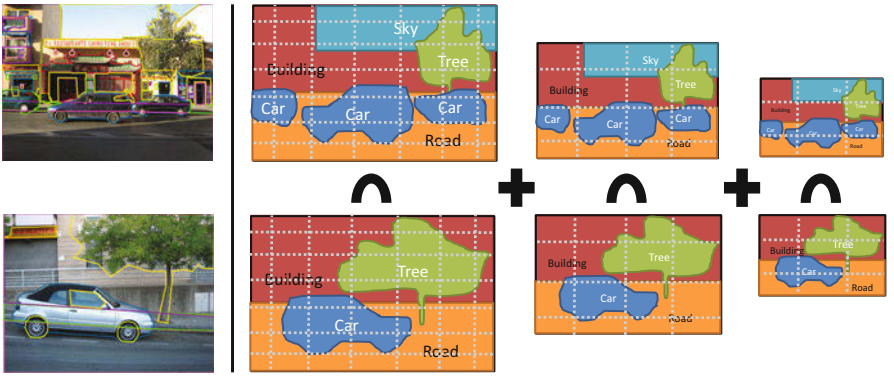


Fig. 7 Left: Where dense annotations are available, they can be used to define a semantic similarity metric between images. In this pair of images from LabelMe, users have labeled pixels as belonging to different objects like cars, roads, tree, sky and so on. Right: The semantic similarity between training images is obtained by computing the intersection of the spatial pyramid histogram built on the object category *label maps* of the two images. The same objects in the same positions produce the highest similarity, and the score degrades smoothly as the locations of the objects differ. See [65] for details.

However, the desired relationships need not be discrete; work in this area also derives continuous semantic *distance* functions from class information or other annotations, specifying a richer set of constraints that ought to be preserved by the binary codes. For example, Fergus *et al.* explore using the distance between classes in WordNet to quantify their semantic distance [20]. Or, rather than enforce similarity only between pairs of classes, one can incorporate a desired similarity between individual images (e.g., by collecting image-level constraints from online annotators).

Additionally, if pixel-level labels exist, as opposed to image-level ones, then more fine-grained measures can be used. Torralba *et al.* [65] define ground truth semantic similarity based a spatial pyramid matching [27, 41] scheme on the object label maps, as illustrated in Figure 7. This results in a simple similarity measure that takes into account the objects present in the image as well as their spatial organization: two images that have the same object labels in similar spatial locations are rated as closer than two images with the same objects but in different spatial locations, and either case is rated closer than two images with different object classes.

3.2 Hash Functions for Learned Mahalanobis Kernels

Having defined possible sources of supervision, we now describe how those semantics are integrated into binary code learning. We first review a hashing-based algorithm for learned Mahalanobis metrics introduced by Jain and colleagues in [37, 40]. The main idea is to learn a parameterization of a Mahalanobis metric (or kernel) based on provided labels or paired constraints for some training examples, while

simultaneously encoding the learned information into randomized hash functions. These functions will guarantee that the more similar inputs are under the learned metric, the more likely they are to collide in a hash table. After indexing all of the database examples with their learned hash keys, those examples similar to a new instance are found in sub-linear time via hashing.

Learned Mahalanobis Metrics and Kernels. The majority of work in metric learning focuses on learning Mahalanobis metrics (e.g., [73, 71, 26, 5, 19]). Given N points $\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\}$, with all $\boldsymbol{x}_i \in \Re^d$, the idea is to compute a positive-definite (p.d.) $d \times d$ matrix A to parameterize the squared Mahalanobis distance:

$$d_A(\boldsymbol{x}_i,\boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T A(\boldsymbol{x}_i - \boldsymbol{x}_j), \tag{2}$$

for all $i, j = 1,\ldots,N$. Note that a generalized inner product (kernel) measures the pairwise similarity associated with that distance:

$$s_A(\boldsymbol{x}_i,\boldsymbol{x}_j) = \boldsymbol{x}_i^T A \boldsymbol{x}_j. \tag{3}$$

The Mahalanobis distance is often used with A as the inverse of the sample covariance when data is assumed to be Gaussian, or with A as the identity matrix if the squared Euclidean distance is suitable.

Let $\mathcal{S}$ and $\mathcal{D}$ denote sets containing pairs of points constrained to be similar and dissimilar, respectively. Given these similarity constraints, one can *learn* the matrix A to yield a measure that is more accurate for a given problem.

For example, Xing *et al.* learn a Mahalanobis metric by using semidefinite programming to minimize the sum of squared distances between similarly labeled examples, while requiring a certain lower bound on the distances between examples with different labels [73]. In related techniques, Globerson and Roweis [24] constrain within-class distances to be zero and maximize between-class distances, Weinberger *et al.* formulate the problem in a large-margin k-nearest-neighbors setting [71], while Goldberger *et al.* maximize a stochastic variant of leave-one-out KNN score on the training set [26]. In addition to using labeled data, research has shown how metric learning can proceed with weaker supervisory information, such as equivalence constraints or relative constraints. For example, equivalence constraints are exploited in the Relevant Component Analysis method of Bar-Hillel *et al.* [5]; the method of Hadsell *et al.* [29] learns a global non-linear mapping of the input data; the Support Vector Machine-based approach of Schultz and Joachims [59] incorporates relative constraints over triples of examples. Davis *et al.* develop an information-theoretic approach that accommodates any linear constraints on pairs of examples, and provide an efficient optimization solution that forgoes eigenvalue decomposition [19].

Main Idea. To use a learned Mahalanobis metric for search, we want to retrieve examples $\boldsymbol{x}_i$ for an input $\boldsymbol{x}_q$ for which the value $d_A(\boldsymbol{x}_i,\boldsymbol{x}_q)$ resulting from Eqn. (2) is small—or, in terms of the kernel form, for which the value of $s_A(\boldsymbol{x}_i,\boldsymbol{x}_q) = \boldsymbol{x}_q^T A \boldsymbol{x}_i$ is high. We next describe how to generate hash functions for the Mahalanobis similarity (1) in the *explicit* case, where the dimensionality of the data is low enough that

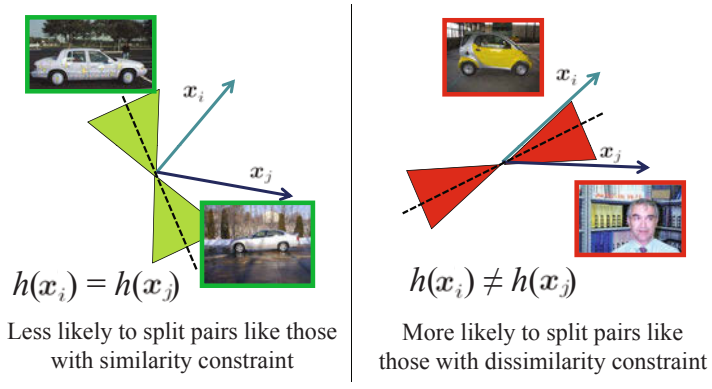


Fig. 8 Whereas traditional unsupervised LSH functions would generate a hyperplane uniformly at random to separate instances, randomized hash functions for a learned kernel are biased so as to ensure similar things become more likely to collide, while dissimilar things become less likely to collide. The hourglass-shaped regions denote that the hash function will be more likely to drawn as such. In the left example, even though the measured angle between x_i and x_j is somewhat wide, the learned hash functions are unlikely to split them into different buckets since the constraints indicate that they (and pairs like them) should be treated as similar.

the $d \times d$ matrix A can be handled in memory, and (2) in the *implicit* case, where A cannot be accessed directly and we want to use a kernelized form of metric learning.

Explicit Formulation. In [12], Charikar proposes a hash function family that is locality-sensitive for the normalized inner product (cosine similarity):

$$\text{sim}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{x}_i^T \boldsymbol{x}_j}{\|\boldsymbol{x}_i\|_2 \|\boldsymbol{x}_j\|_2}. \tag{4}$$

Each hash function simply rounds the output of a product with a random hyperplane:

$$h_{\boldsymbol{r}}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T \boldsymbol{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}, \tag{5}$$

where $\boldsymbol{r}$ is sampled from a zero-mean multivariate Gaussian $\mathcal{N}(0, I)$ of the same dimensionality as the input $\boldsymbol{x}$. The fact that this hash function satisfies the LSH requirement $p[h(\boldsymbol{x}_i) = h(\boldsymbol{x}_j)] = \text{sim}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ relies on a result from Goemans and Williamson [25], who showed that

$$p[\text{sign}(\boldsymbol{x}_i^T \boldsymbol{r}) = \text{sign}(\boldsymbol{x}_j^T \boldsymbol{r})] = 1 - \frac{1}{\pi} \cos^{-1}\left(\frac{\boldsymbol{x}_i^T \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \|\boldsymbol{x}_j\|} \right), \tag{6}$$

for vectors on the unit sphere. This relationship is quite intuitive: the wider the angle between two vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the more likely a randomly selected hyperplane will fall between them, and vice versa.

As shown by Jain *et al.* [37], this is easily extended to accommodate learned Mahalanobis distances. Given the p.d. matrix A, with $A = G^T G$, we generate the following randomized hash functions $h_{\boldsymbol{r},A}$, which accept an input point and return a single hash key bit:

$$h_{\boldsymbol{r},A}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T G\boldsymbol{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}, \tag{7}$$

where the vector $\boldsymbol{r}$ is again chosen at random from a d-dimensional Gaussian distribution with zero mean and unit variance.[4] By parameterizing the hash functions by not only $\boldsymbol{r}$ but also A, we obtain the following relationship:

$$p\left[h_{\boldsymbol{r},A}(\boldsymbol{x}_i) = h_{\boldsymbol{r},A}(\boldsymbol{x}_j)\right] = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{\boldsymbol{x}_i^T A \boldsymbol{x}_j}{\sqrt{|G\boldsymbol{x}_i||G\boldsymbol{x}_j|}}\right),$$

which fulfills the LSH requirement of Eqn. (1) for a metric obtained with any of the Mahalanobis metric learning algorithms. Essentially we have biased the selection of the random hyperplane according to the learned parameters, and by factoring it by G we allow the random hash function itself to "carry" the information about the learned metric. See Figure 8. The denominator in the cosine term normalizes the learned kernel values.

Implicit Formulation. Beyond the explicit formulation given above, we are also interested in the case where the dimensionality d may be very high—say on the order of 10^4 to 10^6—but the examples are sparse and therefore can be stored efficiently. For example, bag of visual word descriptors or histogram pyramids often require millions of dimensions [63, 27, 51]. Even though the examples are themselves sparse and therefore compactly represented, the matrix A can be dense.

In this case, we turn to a particular information-theoretic metric learning (ITML) algorithm developed by Davis *et al.* [19]. In contrast to most other Mahalanobis metric learning approaches, it is kernelizable. It takes an initial "base" parameterization A_0 as input, and then during the learning process it computes *implicit* updates to those parameters, using weighted kernel evaluations between pairs of points involved in the similarity constraints (as opposed to explicit multiplication with A). We briefly summarize the relevant portions of the ITML approach; see [19] for more details.

Information-Theoretic Metric Learning. Given an initial $d \times d$ p.d. matrix A_0 specifying prior knowledge about inter-point distances, the learning task is posed as an optimization problem that minimizes the LogDet loss between A_0 and the ultimate learned parameters A, subject to a set of constraints specifying pairs of examples that are similar or dissimilar (listed in the sets $\mathcal{S}$ and $\mathcal{D}$):

[4] In this case—where A can be explicitly handled in memory—we could equivalently transform all the data according to A *prior* to hashing; however, the choice of presentation here helps set up the formulation presented next.

$$\begin{aligned} \min_{A \succeq 0} \quad & D_{\ell \mathrm{d}}(A, A_0) \\ \text{s. t.} \quad & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq u \quad (i,j) \in \mathcal{S}, \\ & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \ell \quad (i,j) \in \mathcal{D}, \end{aligned} \tag{8}$$

where $D_{\ell \mathrm{d}}(A, A_0) = \mathrm{tr}(AA_0^{-1}) - \log\det(AA_0^{-1}) - d$, d is the dimensionality of the data points, $d_A(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the Mahalanobis distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as defined in Eqn. (2), and ℓ and u are large and small values, respectively.[5] The objective is "information-theoretic" in that it corresponds to minimizing the relative entropy between the associated Gaussians whose covariance matrices are parameterized by A and A_0.

The LogDet loss leads to an efficient algorithm for optimizing Eqn. (8), which involves repeatedly projecting the current solution onto a single constraint, via the explicit update [19]:

$$A_{t+1} = A_t + \beta_t A_t (\boldsymbol{x}_{i_t} - \boldsymbol{x}_{j_t})(\boldsymbol{x}_{i_t} - \boldsymbol{x}_{j_t})^T A_t, \tag{9}$$

where $\boldsymbol{x}_{i_t}$ and $\boldsymbol{x}_{j_t}$ are the constrained data points for iteration t, and β_t is a projection parameter computed (in closed form) by the algorithm.

However, when the dimensionality of the data is very high, one cannot explicitly work with A, and so the update in Eqn. (9) is impractical. Instead, it is replaced with updates in kernel space for an equivalent kernel learning problem in which $K = X^T A X$ for $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_c]$, for a small set of c of the points involved in similarity constraints (see [40]). If K_0 is the input kernel matrix for the data ($K_0 = X^T A_0 X$), then the appropriate update is:

$$K_{t+1} = K_t + \beta_t K_t (\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})^T K_t, \tag{10}$$

where the vectors $\boldsymbol{e}_{i_t}$ and $\boldsymbol{e}_{j_t}$ refer to the i_t-th and j_t-th standard basis vectors, respectively. This update is derived by multiplying Eqn. (9) on the left by X^T and on the right by X. If $A_0 = I$, then the initial kernel matrix is $K_0 = X^T X$; this matrix may be formed using any valid kernel function, and the result of the algorithm is to learn a distance metric on top of this input kernel. By performing the updates in kernel space, the storage requirements change from $O(d^2)$ to $O(c^2)$.

Simultaneous Metric and Hash Function Updates. In order to permit large-scale search with such metrics, the goal is to use the same hash functions as defined above in Eqn. (7), but to express them in a form that is amenable to computing the hash bit with high-dimensional input data. In other words, we want to insert the learned parameters into the hash function and compute $\boldsymbol{r}^T G \boldsymbol{x}$, but now we must do so without working directly with G. To this end, we describe next how to simultaneously make implicit updates to both the hash functions and the metric.

[5] Note that alternatively the constraints may also be specified in terms of relative distances, i.e., $d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) < d_A(\boldsymbol{x}_i, \boldsymbol{x}_k)$. To guarantee the existence of a feasible A, slack variables are also included, but omitted here for brevity.

In [37], Jain *et al.* show how to express G in terms of the initially chosen c data points. Let $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_c]$ be the $d \times c$ matrix of an initial c data points participating in (dis)similarity constraints, and let $\boldsymbol{x}_i^T \boldsymbol{x}_j$ be the initial (non-learned) Mahalanobis similarity value between example $\boldsymbol{x}_i$ and the input $\boldsymbol{x}_j$. Recall the update rule for A from Eqn. (9): $A_{t+1} = A_t + \beta_t A_t \boldsymbol{v}_t \boldsymbol{v}_t^T A_t$, where $\boldsymbol{v}_t = \boldsymbol{y}_t - \boldsymbol{z}_t$, if points $\boldsymbol{y}_t$ and $\boldsymbol{z}_t$ are involved in the constraint under consideration at iteration t. Just as this update must be implemented implicitly via Eqn. (10), so too we must derive an *implicit* update for the G_t matrix required by our hash functions. Since A_t is p.d., we can factorize it as $A_t = G_t^T G_t$, which allows us to rewrite the update as:

$$A_{t+1} = G_t^T (I + \beta_t G_t \boldsymbol{v}_t \boldsymbol{v}_t^T G_t^T) G_t.$$

As a result, factorizing $I + \beta_t G_t \boldsymbol{v}_t \boldsymbol{v}_t^T G_t^T$, we can derive an update for G_{t+1}:

$$\begin{aligned} G_{t+1} &= (I + \beta_t G_t \boldsymbol{v}_t \boldsymbol{v}_t^T G_t^T)^{1/2} G_t \\ &= (I + \alpha_t G_t \boldsymbol{v}_t \boldsymbol{v}_t^T G_t^T) G_t, \end{aligned} \tag{11}$$

where the second equality follows from Lemma 1 in [40] using $\boldsymbol{y} = G_t \boldsymbol{v}_t$, and α_t is defined accordingly.

Using Eqn. (11) and Lemma 2 in [40], G_t can be expressed as $G_t = I + X S_t X^T$, where S_t is a $c \times c$ matrix of coefficients that determines the contribution of each of the c points to G. Initially, S_0 is set to be the zero matrix, and from there every S_{t+1} is iteratively updated in $O(c^2)$ time via

$$S_{t+1} = S_t + \alpha_t (I + S_t K_0)(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})^T (I + K_0 S_t^T)(I + K_0 S_t).$$

Using this result, at convergence of the metric learning algorithm we can compute $G\boldsymbol{x}$ in terms of the c^2 input pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as follows:

$$\begin{aligned} G\boldsymbol{x} &= \boldsymbol{x} + X S X^T \boldsymbol{x} \\ &= \boldsymbol{x} + \sum_{i=1}^{c} \sum_{j=1}^{c} S_{ij} \boldsymbol{x}_j \boldsymbol{x}_i^T \boldsymbol{x}. \end{aligned}$$

Therefore, we have

$$\boldsymbol{r}^T G\boldsymbol{x} = \boldsymbol{r}^T \boldsymbol{x} + \sum_{i=1}^{c} \sum_{j=1}^{c} S_{ij} \boldsymbol{r}^T \boldsymbol{x}_j \boldsymbol{x}_i^T \boldsymbol{x}, \tag{12}$$

and the final implicit hash function $h_{\boldsymbol{r},A}$ for an input $\boldsymbol{x}$ can be defined as:

$$h_{\boldsymbol{r},A}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T \boldsymbol{x} + \sum_{i=1}^{c} \gamma_i^r \boldsymbol{x}_i^T \boldsymbol{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}, \tag{13}$$

where each $\gamma_i^r = \sum_j S_{ij} \boldsymbol{r}^T \boldsymbol{x}_j$.

There are several important things to notice about the ultimate hash function definition. First, we see that the values of each γ_i^r rely only on the basis points, and thus can be efficiently computed in the training phase, prior to hashing anything

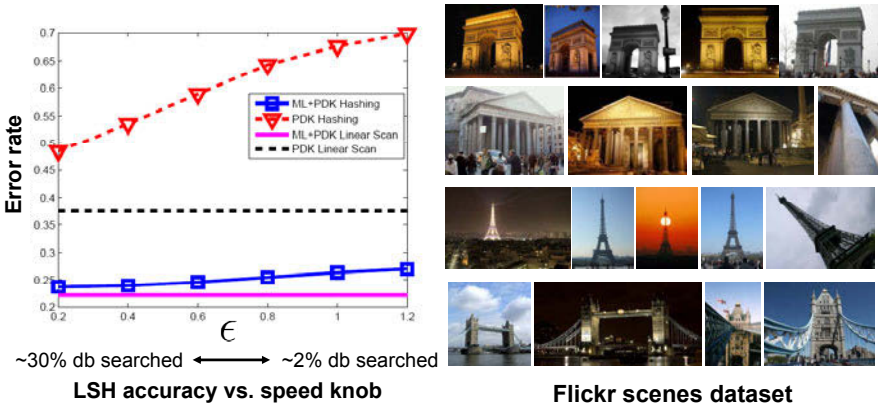


Fig. 9 Hashing error relative to an exhaustive linear scan as a function of ε, which controls the search time required. The error rate of exhaustive search decreases substantially once the similarity constraints on labeled data are used to learn a matching kernel (compare flat dotted black line to flat solid pink line). When these constraints are integrated into the hash function selection, we can obtain similar results to the linear scan, but searching just a fraction of the data (compare ML+PDK Hashing curve to ML+PDK Linear Scan line). Result is from [40].

into the database. Second, the summation consists of as many terms as there are *basis* constrained points c—not the total number of constraints used during metric learning, nor the number of total database points. This is particularly important at query time, when we want the overhead of computing the query's hash key to be low (certainly, it must not require comparing to each database point!) Third, we emphasize that while G is dense and therefore $\boldsymbol{r}^T G$ is not manageable, this method does assume that computing $\boldsymbol{r}^T \boldsymbol{x}$ *is* manageable; for sparse data, only the entries of $\boldsymbol{r}$ corresponding to non-zero entries in $\boldsymbol{x}$ need to be generated.[6] Finally, in practice, it is a strength of the ITML metric learning formulation that one may provide an initial parameterization A_0 based on any *a priori* knowledge, refining it with the external similarity constraints. For example, one could initialize with a pyramid match kernel, and refine it with the similarity constraints.

Figure 9 shows an example result using these supervised LSH functions to index Flickr images of 18 different tourist sites, where the images are represented with sets of local SIFT [45] features. Using a base matching kernel [43] as input, the algorithm learns a Mahalanobis kernel on top of it, and simultaneously updates the LSH function parameterization. Then a nearest-neighbor scene classification task is posed. The results illustrate the nice control one has on the balance between (i) the search time required and (ii) the accuracy guaranteed relative to a linear scan, as determined by the LSH parameter ε (discussed above in Section 2.3). Searching only about 2% of the database, we see error levels similar to that of an exhaustive linear scan with the learned metric.

[6] Thus it is particularly efficient when the inputs are sparse. If they are high-dimensional but dense, the implicit form is still valuable, as it bypasses computing $O(d^2)$ products with G and requires only $O(d)$ inner products for $\boldsymbol{r}^T \boldsymbol{x}$.

Interestingly, this plot also reveals that the learning stage has a dimensionality reduction effect. While both hashing curves use the same number of hash bits b, the learned hash functions more closely approximate the associated linear scan result. (Compare the relative gaps between the top two curves and the bottom two curves, respectively.) We can attribute this to the fact that the learned hash functions usefully focus the partitions' placement in the original feature space, requiring fewer bits for the same representational power.

3.3 Learned Binary Embeddings for Semantic Similarity

The previous section reviewed methods for hashing with learned Mahalanobis metrics, where the similarity constraints are used to establish a linear transformation of the original input space (possibly implicitly computed). In this section we consider other more general types of transformations that can be constructed using labeled training examples. The main idea is to exploit supervised learning algorithms to aggregate a set of functions that jointly preserve the desired neighborhood structure. Whereas the learned kernel technique above generates locality-sensitive hash functions, these techniques generate Hamming embeddings (refer back to Section 2 for contrasts).

Specifically, the methods in this section address the following learning problem: given a database of images $\{\boldsymbol{x}_i\}$ and a distance function $d(\boldsymbol{x}_i,\boldsymbol{x}_j)$ we seek a binary feature vector $\boldsymbol{y}_i = f(\boldsymbol{x}_i)$ that preserves the nearest neighbor relationships using a Hamming distance. Formally, for a point $\boldsymbol{x}_i$, denote by $\mathcal{N}_{100}(\boldsymbol{x}_i)$ the indices of the 100 nearest neighbors of $\boldsymbol{x}_i$ according to a semantic distance function $d(\boldsymbol{x}_i,\boldsymbol{x}_j)$ derived using one of the supervision forms described in Section 3.1. Similarly, define $\mathcal{N}_{100}(\boldsymbol{y}_i)$ to be the set of indices of the 100 descriptors $\boldsymbol{y}_j$ that are closest to $\boldsymbol{y}_i$ in terms of Hamming distance. Ideally, we would like $\mathcal{N}_{100}(\boldsymbol{x}_i) = \mathcal{N}_{100}(\boldsymbol{y}_i)$ for all examples in our training set.

We discuss two general approaches to learning the explicit mapping functions. The first is a variant of the *Parameter Sensitive Hashing* (PSH) algorithm of Shakhnarovich *et al.*, which uses boosting and a rounding-based LSH function to select feature dimensions that are most indicative of similarity in some parameter space of interest (e.g., human pose joint angles in their application) [61, 60]. The second is a neural network-based approach explored by Salakhutdinov and Hinton [57] and Torralba *et al.* [65], the former being used for document retrieval. These models utilize a form of unsupervised pre-training using a stack of restricted Boltzmann machines (RBMs).

3.3.1 Boosting-Based Embedding

In Shakhnarovich *et al.* [61], each image is represented by a binary vector with b bits $\boldsymbol{y}_i = [h_1(\boldsymbol{x}_i), h_2(\boldsymbol{x}_i), ..., h_b(\boldsymbol{x}_i)]$, so that the distance between two images is given by a weighted Hamming distance $d(\boldsymbol{x}_i,\boldsymbol{x}_j) = \sum_{l=1}^{b} \alpha_l |h_l(\boldsymbol{x}_i) - h_l(\boldsymbol{x}_j)|$. The weights

α_i and the functions $h_l(\boldsymbol{x}_i)$ are binary regression stumps that map the input vector $\boldsymbol{x}_i$ into binary features and are learned using Boosting.

For the learning stage, positive examples are pairs of images $\boldsymbol{x}_i, \boldsymbol{x}_j$ so that $\boldsymbol{x}_j$ is one of the nearest neighbors of $\boldsymbol{x}_i$, $j \in \mathcal{N}(\boldsymbol{x}_i)$. Negative examples are pairs of images that are not neighbors. In our implementation we use GentleBoost with regression stumps to minimize the exponential loss. In PSH, each regression stump has the form:

$$f_l(\boldsymbol{x}_i, \boldsymbol{x}_j) = \alpha_l \left[(e_l^T \boldsymbol{x}_i > T_l) = (e_l^T \boldsymbol{x}_j > T_l) \right] + \beta_l. \tag{14}$$

At each iteration l, we select the parameters of f_l, the regression coefficients (α_l, β_l), the stump parameters (where e_l is a unit vector, so that $e_l^T \boldsymbol{x}$ returns the lth component of $\boldsymbol{x}$, and T_l is a threshold), to minimize the square loss:

$$\sum_{n=1}^{N} w_l^n (z_n - f_l(\boldsymbol{x}_i^n, \boldsymbol{x}_j^n))^2, \tag{15}$$

where N is the number of training pairs, z_n is the neighborhood label ($z_n = 1$ if the two images are neighbors and $z_n = -1$ otherwise), and w_l^n is the weight for each training pair at iteration l given by $w_l^n = \exp(-z_n \sum_{t=1}^{l-1} f_l(\boldsymbol{x}_i^n, \boldsymbol{x}_j^n))$.

In Torralba *et al.* [65] the authors constrain the metric to be a Hamming distance, restricting the class of weak learners so that all the weights are the same for all the features $\alpha_l = \alpha$. (The values of β_l do not need to be constrained as they only contribute to final distance as a constant offset, independent of the input pair.) This small modification is important as it permits standard Hashing techniques to be used. The parameter α has an effect in the generalization of the final function.

Once the learning stage is finished, every image can be compressed into b bits, where each bit is computed as $h_l(\boldsymbol{x}_i) = e_l^T \boldsymbol{x}_i > T_l$. The algorithm is simple to code, and relatively fast to train.

3.3.2 Restricted Boltzmann Machines-Based Embedding

The second approach uses the dimensionality reduction framework of Salakhutdinov and Hinton [32, 57], based on multiple layers of restricted Boltzmann machines (RBMs). We first give a brief overview of RBM's, before describing their use in Torralba *et al.* [65] where they are applied to images.

An RBM models an ensemble of binary vectors with a network of stochastic binary units arranged in two layers, one visible, one hidden. Units $\mathbf{v}$ in the visible layers are connected via a set of symmetric weights W to units $\mathbf{h}$ in the hidden layer. The joint configuration of visible and hidden units has an energy:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i \in \text{visible}} b_i v_i - \sum_{j \in \text{hidden}} b_j h_j - \sum_{i,j} v_i h_i w_{ij}, \tag{16}$$

where v_i and h_j are the binary states of visible and hidden units i and j. The weights are denoted by w_{ij}, and b_i and b_j are bias terms, also model parameters. Using this energy function, a probability can be assigned to a binary vector at the visible units:

$$p(\mathbf{v}) = \sum_{\mathbf{h}} \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{u},\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})}}. \tag{17}$$

RBMs lack connections between units within a layer, hence the conditional distributions $p(\mathbf{h}|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h})$ have a convenient form, being products of Bernoulli distributions:

$$\begin{aligned} p(h_j = 1|\mathbf{v}) &= \sigma(b_j + \sum_i w_{ij} v_i) \\ p(v_i = 1|\mathbf{h}) &= \sigma(b_i + \sum_j w_{ij} h_j), \end{aligned} \tag{18}$$

where $\sigma(u) = 1/(1+e^{-u})$, the logistic function. Using Eqn. 18, parameters w_{ij}, b_i, b_j can be updated via a contrastive divergence sampling scheme (see [32] for details). This ensures that the training samples have a lower energy than nearby hallucinations, samples generated synthetically to act as negative examples.

Hinton and colleagues have demonstrated methods for stacking RBMs into multiple layers, creating "deep" networks which can capture high order correlations between the visible units at the bottom layer of the network. By choosing an architecture that progressively reduces the number of units in each layer, a high-dimensional binary input vector can be mapped to a far smaller binary vector at the output. Thus each bit at the output maps through multiple layers of non-linearities to model the complicated subspace of the input data.

Since the input descriptors will typically be real-valued, rather than binary (e.g. Gist or SIFT descriptors), the first layer of visible units are modified to have a Gaussian distribution.[7]

The deep network is trained into two stages: first, an unsupervised *pre-training* phase which sets the network weights to approximately the right neighborhood; second, a *fine-tuning* phase where the network has its weights moved to the local optimum by back-propagation on labeled data.

In pre-training, the network is trained from the visible input layer up to the output layer in a greedy fashion. Once the parameters of the first layer have converged using contrastive divergence, the activation probabilities (given in Eqn. 18) of the hidden layer are fixed and used as data for the layer above—the hidden units becoming the visible ones for the next layer up, and so on up to the top of the network.

In fine-tuning, the units are made deterministic, retaining the weights and biases from pre-training and performing gradient descent on them using back-propagation. One possible objective function is Neighborhood Components Analysis (NCA) [26, 56]. This attempts to preserve the semantic neighborhood structure by maximizing the number of neighbors around each query that have the same class labels. Given N labeled training cases $(\mathbf{x}^n, c^n)$, denote the probability that point n is assigned the

[7] In Eqn. 18, $p(v_i = u|\mathbf{h})$ is modified to be a Gaussian with a mean determined by the hidden units; see [56].

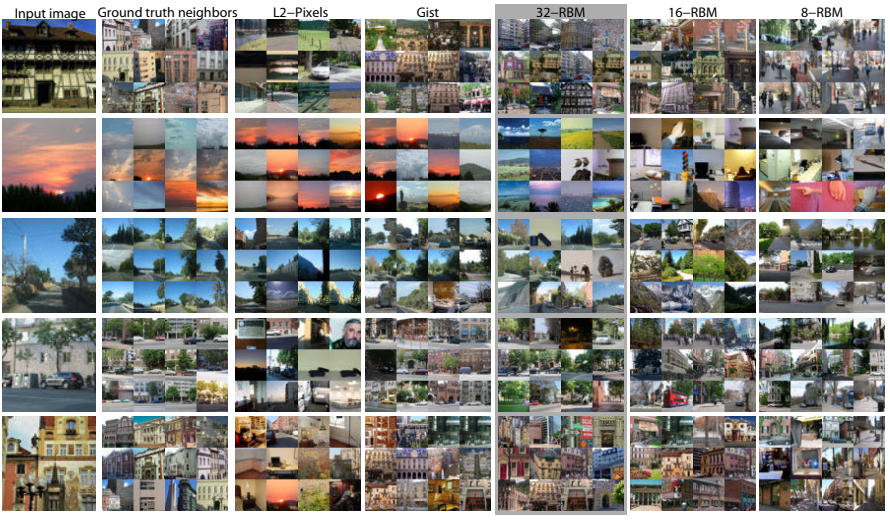


Fig. 10 Each row shows the input image and the 12 nearest neighbors using, a) ground truth distance using the histograms of objects present on each image (see text), b) L2 distance using RGB values, c) L2 distance using Gist descriptors, d) Gist features compressed to 8 bits using an RBM and Hamming distance, e) 16 bits RBM and f) 32 bits RBM.

class of point m as p_{nm}. The objective O_{NCA} attempts to maximize the expected number of correctly classified points on the training data:

$$O_{\text{NCA}} = \sum_{n=1}^{N} \sum_{l:c^n=c^l} p_{nm}, \; p_{nm} = \frac{e^{-||f(\mathbf{x}^n|W)-f(\mathbf{x}^l|W)||^2}}{\sum_{m\neq l} e^{-||f(\mathbf{x}^m|W)-f(\mathbf{x}^l|W)||^2}},$$

where $f(\mathbf{x}|W)$ is the projection of the data point $\mathbf{x}$ by the multi-layered network with parameters W. This function can be minimized by taking derivatives of O_{NCA} with respect to W and using conjugate gradient descent. Alternative objective functions include the DrLIM objective introduced by Hadsell *et al.* [29].

Figure 10 shows representative retrieval results on a 20,000 LabelMe dataset. Gist descriptors [50] are used as the high-dimensional input representation for each image (single descriptor per image). Figure 11 provides a quantitative analysis of the retrieval performance on 2,000 test images. Figure 11(a) displays the percentage of the first true 50 nearest neighbors that are included in the retrieved set as a function of the number of the images retrieved (M). Figure 11(b) shows a section of Figure 11(a) for 500 retrieved images. The figures compare LSH (with no learning), PSH and RBMs. Figure 11(b) shows the effect of increasing the number of bits. Top performance is reached with around 30 bits for RBMs, with the other methods requiring more bits. However, given enough bits, all the approaches converge to similar retrieval performance. The matching speed using the binary codes and the Gist descriptors is shown in Figure 2, where the compact codes facilitate extremely fast retrieval.

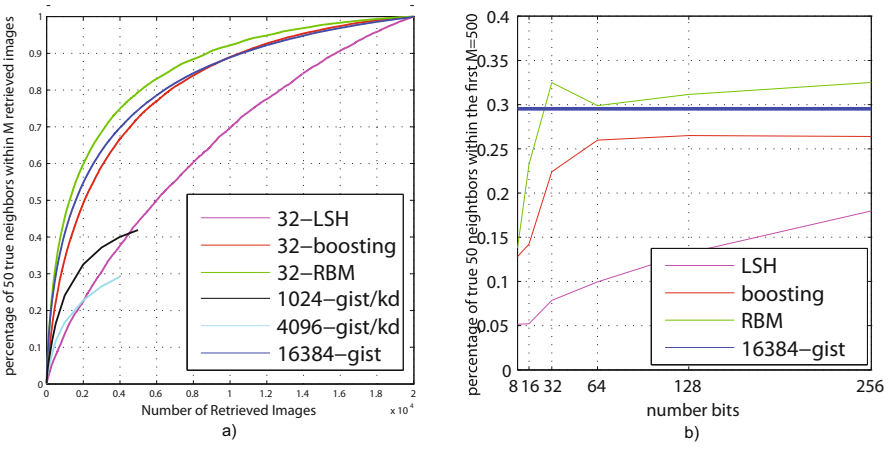


Fig. 11 **(a):** For a fixed number of true neighbors ($|\mathcal{N}| = 50$), we plot the percentage of true nearest neighbors that are retrieved as a function of the total number of images retrieved. True neighbors are defined in terms of object label histograms (see Figure 7). The original Gist descriptors (blue) perform well but a slow to match due to high dimensionality. LSH (magenta), which does not use learning, performs less well than Boosting (red) and the RBM-based (green) methods. The Boosting and RBM-based embeddings, despite using only 32-bits per descriptor match the performance of the original Gist. **(b):** Varying the number of bits for 500 retrieved images.

Figure 11(a) also shows a comparison with the more conventional *kd*-tree based methods. Here the FLANN [47] *kd*-tree implementation was applied to the Gist descriptors (converted to uint8), both with (black) and without (cyan) a preliminary PCA projection down to 128 dimensions. To give a fair comparison, the *kd*-tree parameters were adjusted to give a comparable retrieval time to the other methods. The performance can be seen to be considerably worse than the approaches using binary codes. This is due to the poor performance of *kd*-tree type approaches in high dimensional spaces.

3.4 Other Supervised Methods

Building on the ideas presented thus far, recent work has explored alternative methods to learn hash functions. Wang *et al.* propose a supervised form of PCA, where pairwise binary labels act as constraints on the projection [70]. Projecting new examples with this approach requires a Nystrom-based out-of-sample projection. Mu and colleagues develop a kernel-based maximum margin approach to select hash functions [46], and a semi-supervised approach that minimizes empirical error on a labeled constraint set while promoting independence between bits and balanced partitions is described in [69]. The SPEC hashing approach [42] uses a conditional

entropy measure to add binary functions in a way that matches a desired similarity function, but approximately so as to ensure linear run-time.

Jain and colleagues develop a dynamic hashing idea to accommodate metrics learned in an online manner, where similarity constraints are accumulated over time rather than made available at once in batch [36]. Bronstein and colleagues extend the idea of learning binary similarity functions with boosting to the cross-modal case, where data comes from two different input spaces (e.g., we want to judge the similarity between a CT and a PET image) [11].

Whereas the technique in Section 3.2 above connects Mahalanobis and ITML-learned kernels to LSH, the Kernelized LSH approach developed by Kulis & Grauman provides a connection for *arbitrary* kernel functions [39], which includes kernels learned with ITML or otherwise. We will review this method in Section 4.3, since it can be applied in both supervised and unsupervised settings.

4 Unsupervised Methods for Defining Binary Projections

In the previous section, we reviewed supervised algorithms that require some form of label information to define which points should be close by and which should be far apart in the binary space. We now look at unsupervised techniques that simply try to preserve the neighborhood structure between points in the input space, and thus require no labels.

The goal is to compute a binary representation of the original representation for each image, so that similar instances have similar binary codes. Typically the original feature space and distance are Euclidean, but alternatives are possible, as discussed in Sections 4.1 and 4.3 below. Additionally, we want the code to be easily computed for a novel input and to be compact in length, thus enabling efficient methods such as Semantic Hashing (see Section 2.2) to be used.

We first briefly summarize several methods for specific similarity functions (Section 4.1), then discuss a spectral approach for coding the Euclidean distance on real-valued vector data (Section 4.2), and finally review an approach to generate codes for kernel functions, including those over non-vector data (Section 4.3).

4.1 Binary Codes for Specific Similarity Functions

Several embedding functions that map a specialized distance into a generic space (e.g., Euclidean) have been developed to exploit either hashing or Hamming space search for particular metrics of interest. In order to exploit known LSH functions [17], Indyk and Thaper design a low-distortion L_1 embedding for the bijective match distance between two sets of feature vectors [34]. Grauman and Darrell construct a related embedding for the normalized partial match, showing that an implicit unary encoding with a linear kernel is equivalent to the pyramid match kernel on feature sets [28], thereby allowing hashing with the function in Eqn. (5). A related embedding is given in [40] for the proximity distribution kernel [43],

which is an image matching kernel that accounts for both the correspondence between features and their relative spatial layout.

The Min-Hash technique introduced by Broder [10] is a randomized embedding designed to capture the normalized set overlap: $\text{sim}(S_1, S_2) = \frac{|S_1 \bigcap S_2|}{|S_1 \bigcup S_2|}$, for sets S_1 and S_2. Assuming we have a discrete set (vocabulary) of tokens that may appear in any set, we generate a random permutation of those unique tokens. Then, given any input set, its Min-Hash key is the token present in the set that has the minimum value (appears first) in the permutation. The probability that two sets receive the same Min-Hash value is equal to the overlap similarity. Intuitively, the permutations correspond to picking some token from the sets' union at random, and checking whether both contain it. The higher the overlap between two sets, the more likely we draw a token they share. Similar to the hash table construction process described in Section 2.3, one concatenates multiple such hash values to generate a hash key, and aggregates results over multiple independently generated functions. The set overlap is meaningful for gauging document similarity, and Min-Hash was initially used for efficient near-duplicate detection among Web pages. Chum and colleagues have shown its suitability for near-duplicate detection for bag of words image representations as well, and adapt the idea to include the effective tf-idf weighting [14], and to integrate spatial layout of features into the hash selection [13].

Notably, all of the above projection techniques cater to *sets* of features, where each instance is comprised of some variable number of descriptors, and the desired distance computes some matching or overlap between them. Such techniques' success for image search applications is a result of the strong *local feature* representations used widely in the recognition and CBIR communities in the last decade.

Aside from set-based metrics, Rahimi and Recht design embeddings for a particular form of shift-invariant kernel. They propose randomized mappings into a real-valued low-dimensional feature space such that an inner product approximates a given shift-invariant kernel, such as the Gaussian or Laplacian kernel [53]. Note that while the intent in that work is to exploit linear machine learning algorithms that permit fast training with large-scale data (as opposed to search), one could take such an embedding and again use the randomized hyperplane hash functions (Eqn. (5)). Raginsky and Lazebnik also show how to convert those real-valued mappings to binary outputs so that Hamming space search is applicable, with bounds on the expected normalized Hamming distance relative to the original shift-invariant kernel value [52].

4.2 Spectral Hashing

Whereas the above section addresses unsupervised codes developed for particular similarity functions of interest, we now examine a technique that not only aims to preserve the given similarities, but also attempts to satisfy generic properties that make compact binary codes effective. This is the Spectral Hashing framework developed by Weiss *et al.* [72].

In formalizing the requirements for a good code, we see that they are equivalent to a particular form of graph partitioning. This means that even for a single bit, the

problem of finding optimal codes is NP hard. On the other hand, the analogy to graph partitioning suggests a relaxed version of the problem that leads to very efficient eigenvector solutions. These eigenvectors are exactly the eigenvectors used in many spectral algorithms including spectral clustering and Laplacian eigenmaps [6, 49], hence the name "Spectral Hashing" [72].

We have already discussed several basic requirements of a good binary code: it should (1) be easily computed for a novel input; (2) require a small number of bits to code the full dataset and (3) map similar items to similar binary codewords. Beyond these basic requirements, however, the Spectral Hashing approach also aims to form codes that more efficiently utilize each bit. Specifically, we require that each bit have a 50% chance of being one or zero, and that different bits be independent of each other. Among all codes that have this property, we will seek the ones where the average Hamming distance between similar points is minimal.

Let $\{\boldsymbol{y}_i\}_{i=1}^N$ be the list of codewords (binary vectors of length b) for N data points and $W_{N\times N}$ be the affinity matrix. Assuming the inputs are embedded in R^d so that Euclidean distance correlates with similarity, a suitable affinity is $W(i,j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\varepsilon^2)$. Thus the parameter ε defines the distance in R^d which corresponds to similar items. Using this notation, the average Hamming distance between similar neighbors can be written: $\sum_{ij} W_{ij}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2$. By relaxing the independence assumption and requiring the bits to be *uncorrelated* the following problem is obtained:

$$\begin{aligned} \textit{minimize} : & \sum_{ij} W_{ij}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 \\ \textit{subject to} : & \boldsymbol{y}_i \in \{-1,1\}^b \\ & \sum_i \boldsymbol{y}_i = 0 \\ & \frac{1}{N}\sum_i \boldsymbol{y}_i \boldsymbol{y}_i^T = I, \end{aligned} \tag{19}$$

where the constraint $\sum_i \boldsymbol{y}_i = 0$ requires each bit to fire 50% of the time, and the constraint $\frac{1}{N}\sum_i \boldsymbol{y}_i \boldsymbol{y}_i^T = I$ requires the bits to be uncorrelated. For a single bit, solving problem 19 is equivalent to balanced graph partitioning and is NP hard (see [72] for proof), thus the problem must be relaxed in some way to make it tractable.

Spectral Relaxation. By introducing an $N \times b$ matrix Y whose jth row is $\boldsymbol{y}_j^T$ and a diagonal $N \times N$ matrix $D(i,i) = \sum_j W(i,j)$, the problem can be rewritten as:

$$\begin{aligned} \textit{minimize} : & \textit{trace}(Y^T(D-W)Y) \\ \textit{subject to} : & Y(i,j) \in \{-1,1\} \\ & Y^T 1 = 0 \\ & Y^T Y = I \end{aligned} \tag{20}$$

This is of course still a hard problem, but by removing the constraint that $Y(i,j) \in \{-1,1\}$ an easier problem is obtained whose solutions are simply the b eigenvectors

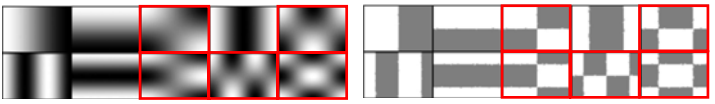


Fig. 12 Left: Eigenfunctions for a uniform rectangular distribution in 2D. Right: Thresholded eigenfunctions. Outer-product eigenfunctions have a red frame. The eigenvalues depend on the aspect ratio of the rectangle and the spatial frequency of the cut—it is better to cut the long dimension first, and lower spatial frequencies are better than higher ones.

of $D - W$ with minimal eigenvalue (after excluding the trivial eigenvector 1 which has eigenvalue 0).

Out-of-Sample Extension. The out-of-sample extension of spectral methods is often solved using the Nystrom method [8, 21]. However, note that the cost of calculating the Nystrom extension of a new datapoint is *linear* in the size of the dataset. With millions of items in the dataset this is impractical. In fact, calculating the Nystrom extension is as expensive as doing exhaustive nearest neighbor search.

In order to enable an efficient out-of-sample extension, the data points $\boldsymbol{x}_i \in R^d$ are assumed to be samples from a probability distribution $p(\boldsymbol{x})$. The equations in problem 19 above are now seen to be sample averages, which can be replaced by their expectations:

$$\begin{aligned} \textit{minimize}: \int \|\boldsymbol{y}(\boldsymbol{x}_1) - \boldsymbol{y}(\boldsymbol{x}_2)\|^2 W(\boldsymbol{x}_1, \boldsymbol{x}_2) p(\boldsymbol{x}_1) p(\boldsymbol{x}_2) d\boldsymbol{x}_1 \boldsymbol{x}_2 \\ \textit{subject to}: \boldsymbol{y}(\boldsymbol{x}) \in \{-1, 1\}^b \\ \int \boldsymbol{y}(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} = 0 \\ \int \boldsymbol{y}(\boldsymbol{x}) \boldsymbol{y}(\boldsymbol{x})^T p(\boldsymbol{x}) d\boldsymbol{x} = I, \end{aligned} \tag{21}$$

with $W(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2/\varepsilon^2}$. Relaxing the constraint that $\boldsymbol{y}(\boldsymbol{x}) \in \{-1, 1\}^b$ results in a spectral problem whose solutions are *eigenfunctions* of the weighted Laplace-Beltrami operators defined on manifolds [15, 7, 8, 48].

What do the eigenfunctions $\Psi_b(\boldsymbol{x})$ look like? One important special case is when $p(\boldsymbol{x})$ is a separable distribution. A simple case of a separable distribution is a multidimensional uniform distribution $p(\boldsymbol{x}) = \prod_i u_i(\boldsymbol{x}_i)$ where u_i is a uniform distribution in the range $[a_i, \bar{a}_i]$. In the uniform case, the eigenfunctions $\Psi_b(\boldsymbol{x})$ and eigenvalues λ_b are:

$$\Psi_b(\boldsymbol{x}) = \sin\left(\frac{\pi}{2} + \frac{b\pi}{\bar{a} - a}\boldsymbol{x}\right) \tag{22}$$

$$\lambda_b = 1 - e^{-\frac{\varepsilon^2}{2}\left|\frac{b\pi}{\bar{a}-a}\right|^2}. \tag{23}$$

Figure 12 shows the analytical eigenfunctions for a 2D rectangle in order of increasing eigenvalue. The eigenvalue (which corresponds to the cut) determines which b bits will be used. Note that the eigenvalue depends on the aspect ratio of the

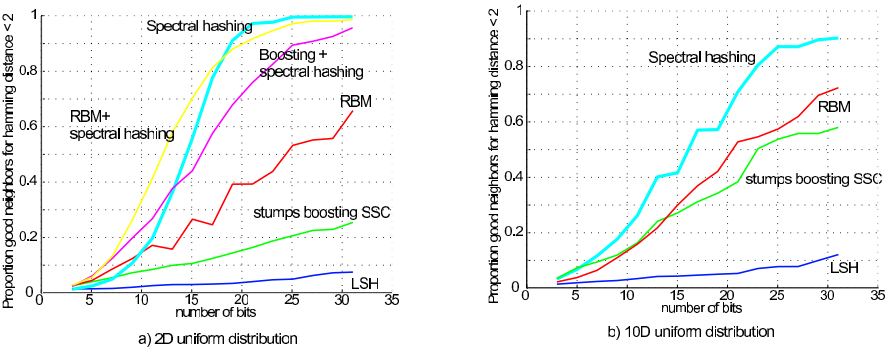


Fig. 13 Left: results on 2D rectangles with different methods. Even though Spectral Hashing is the simplest, it gives the best performance. Right: Similar pattern of results for a 10-dimensional distribution.

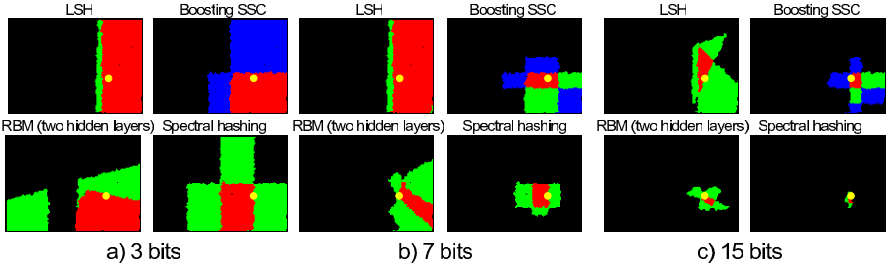


Fig. 14 Comparison of neighborhood defined by Hamming balls of different radii using codes obtained with vanilla LSH, Boosting, RBM, and Spectral Hashing when using 3, 7 and 15 bits. The yellow dot denotes a test sample. The red points correspond to the locations that are within a Hamming distance of zero. Green corresponds to a Hamming ball of radius 1, and blue to radius 2.

rectangle and the spatial frequency—it is better to cut the long dimension before the short one, and low spatial frequencies are preferred.

We distinguish between *single-dimension* eigenfunctions, which are of the form $\Psi_b(\boldsymbol{x}_1)$ or $\Psi_b(\boldsymbol{x}_2)$ and *outer-product* eigenfunctions which are of the form $\Psi_b(\boldsymbol{x}_1)\Psi_l(\boldsymbol{x}_2)$. These outer-product eigenfunctions are shown marked with a red border in the figure. As we discuss below, these outer-product eigenfunctions should be avoided when building a hashing code.

Summary of Algorithm. Recapping, given a training set of points $\{\boldsymbol{x}_i\}$ and a desired number of bits b, the steps of the Spectral Hashing algorithm are:

- Find the principal components of the data using PCA.
- Calculate the b smallest *single-dimension* analytical eigenfunctions of L_p using a rectangular approximation along every PCA direction. This is done by evaluating

the b smallest eigenvalues for each direction using (Eqn. 22), thus creating a list of db eigenvalues, and then sorting this list to find the b smallest eigenvalues.
- Threshold the analytical eigenfunctions at zero, to obtain binary codes.

Illustrative Results. Figure 13(a) shows a comparison between Spectral Hashing and Euclidean LSH, RBMs, and Boosting on a 2D rectangle of data. Despite the simplicity of Spectral Hashing, it outperforms the other methods. Indeed, even when we apply RBMs and Boosting to the output of Spectral Hashing the performance does not improve. A similar pattern of results is shown for a 10D rectangle (Figure 13(b)). Note that the Boosting and RBM methods were trained using the approach described in Section 3.3.1 and Section 3.3.2, respectively, but using a distance matrix $D = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\varepsilon^2)$, instead of one produced by supervised label information.

Some insight into the superior performance can be gained by comparing the partitions that each bit defines on the data (see Figure 12). Recall that we seek partitions that give low cut value and are approximately independent. If simply using random linear partitions, LSH can give very unbalanced partitions. RBMs and Boosting both find good partitions, but the partitions can be highly dependent on each other. Spectral Hashing finds well balanced partitions that are more compact than those of the other methods, showing it makes efficient use of a given number of bits.

Figure 15 shows retrieval results for Spectral Hashing, RBMs, and Boosting on the LabelMe dataset [65], using Gist descriptors as the input. Note that even though Spectral Hashing uses a poor model of the statistics of the database—it simply assumes a N-dimensional rectangle, it performs better than Boosting which actually uses the distribution (the difference in performance relative to RBMs is not significant). Not only is the performance numerically better, but our visual inspection of the retrieved neighbors suggests that with a small number of bits, the retrieved images are better using Spectral Hashing than with Boosting. However, Spectral Hashing can only emulate the distance between Gist descriptors, as it has no mechanism for using label information, whereas Boosting or RBMs do (see Section 3).

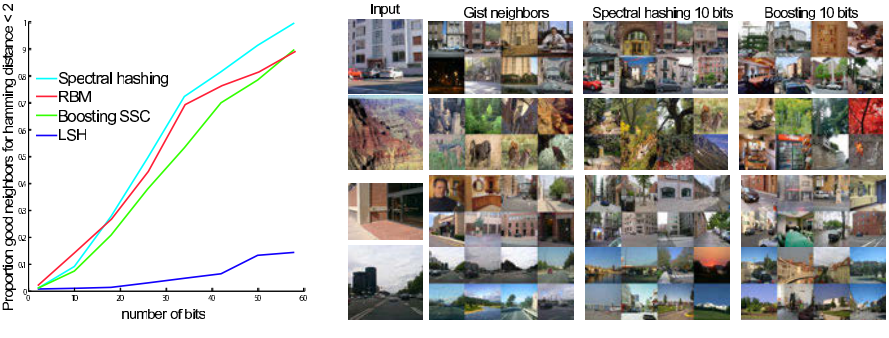


Fig. 15 Performance of different binary codes on the LabelMe dataset described in [65]. The data is certainly not uniformly distributed, and yet Spectral Hashing gives better retrieval performance than Boosting or vanilla LSH.

4.3 Kernelized Locality Sensitive Hashing

Kernel functions are a valuable family of similarity measures, particularly since they can support structured input spaces (sets, graphs, trees) and enable connections with kernel learning algorithms. However, methods discussed thus far either assume that the data to be hashed comes from a multidimensional vector space, or require that the underlying embedding of the data be explicitly known and computable. For example, Spectral Hashing assumes uniformly distributed data in R^d; the random hyperplane LSH function expects vector data [12]; and certain specialized embeddings are manually crafted for a function of interest (Section 4.1).

This is limiting, given that many recent successful vision results employ kernels for which the underlying embedding is known only *implicitly* (i.e., only the kernel function is computable). This includes various kernels designed specifically for image comparisons (e.g., [76, 77, 68]), as well as some basic widely used functions like a Gaussian RBF kernel, or arbitrary (e.g., non-Mahalanobis) learned kernels.

Therefore, we next overview the *kernelized locality-sensitive hashing* (KLSH) approach recently introduced by Kulis and Grauman [39], which shows how to construct randomized locality-sensitive functions for arbitrary kernel functions. KLSH generalizes LSH to scenarios when the kernel-induced feature space embedding is either unknown or incomputable.

Main Idea. Formally, given an arbitrary (normalized) kernel function κ, we have

$$\text{sim}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sqrt{\kappa(\boldsymbol{x}_i, \boldsymbol{x}_i)\kappa(\boldsymbol{x}_j, \boldsymbol{x}_j)}} \tag{24}$$

$$= \frac{\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)}{\|\phi(\boldsymbol{x}_i)\|_2 \|\phi(\boldsymbol{x}_j)\|_2}, \tag{25}$$

for some (possibly unknown) embedding function $\phi(\cdot)$. As usual, given a database of n objects, the goal is to quickly find the most similar item to a query object $\boldsymbol{q}$ in terms of the kernel function, that is, $\text{argmax}_i \kappa(\boldsymbol{q}, \boldsymbol{x}_i)$. Since we know that any Mercer kernel can be written as an inner product in some high-dimensional space [62], at a glance we might consider simply employing the random hyperplane hash functions introduced earlier in Eqn. (5), which is locality-sensitive for the inner product.

However, looking more closely, it is unclear how to do so. The random hyperplane projections assume that the vectors are represented explicitly, so that the sign of $\boldsymbol{r}^T \boldsymbol{x}$ can easily be computed. That would require referencing a random hyperplane *in the kernel-induced feature space*, but we have access to the data only through the kernel function $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$. For example, the RBF kernel has an infinite-dimensional embedding, making it seemingly impossible to construct $\boldsymbol{r}$. Thus the key challenge in applying LSH to this scenario is in constructing a vector $\boldsymbol{r}$ from $\mathcal{N}(0, I)$ such that $\boldsymbol{r}^T \phi(\boldsymbol{x})$ can be computed via the kernel function.

The main idea of KLSH is to construct $\boldsymbol{r}$ as a weighted sum of a subset of the database items, drawing on the central limit theorem. In doing so, like standard LSH, hash functions are computed as random projections; however, unlike standard LSH,

these random projections will be constructed using only the kernel function and a sparse set of representative examples.

Algorithm. Consider each data point $\phi(\boldsymbol{x}_i)$ from the database as a vector from some underlying distribution $\mathcal{D}$ with mean μ and covariance Σ, which are generally unknown. Given a natural number t, define

$$\boldsymbol{z}_t = \frac{1}{t}\sum_{i\in S}\phi(\boldsymbol{x}_i), \tag{26}$$

where S is a set of t database objects chosen i.i.d. from $\mathcal{D}$. According to the central limit theorem [54], for sufficiently large t, the random vector

$$\tilde{\boldsymbol{z}}_t = \sqrt{t}(\boldsymbol{z}_t - \mu) \tag{27}$$

is distributed according to the multi-variate Gaussian $\mathcal{N}(0,\Sigma)$. By applying a whitening transform, the vector $\Sigma^{-1/2}\tilde{\boldsymbol{z}}_t$ will be distributed according to $\mathcal{N}(0,I)$, precisely the distribution required for hashing.

Therefore, we denote our random vector as $\boldsymbol{r} = \Sigma^{-1/2}\tilde{\boldsymbol{z}}_t$, and the desired hash function $h(\phi(\boldsymbol{x}))$ is given by

$$h(\phi(\boldsymbol{x})) = \begin{cases} 1, & \text{if } \phi(\boldsymbol{x})^T\Sigma^{-1/2}\tilde{\boldsymbol{z}}_t \geq 0 \\ 0, & \text{otherwise} \end{cases}. \tag{28}$$

Now the issue becomes how to express the product of the implicit random vector $\tilde{\boldsymbol{z}}_t$ and the matrix $\Sigma^{-1/2}$ as a weighted sum of kernel-space instances.

To do this, KLSH uses a technique similar to that used in kernel Principal Component Analysis (kPCA) [58] to project onto the eigenvectors of the covariance matrix, as follows. Both the covariance matrix Σ and the mean μ of the data are unknown, and must be approximated via a sample of the data. We choose a set of p database objects, which we denote without loss of generality as the first p items $\phi(\boldsymbol{x}_1),\ldots,\phi(\boldsymbol{x}_p)$ of the database (where $p \ll n$), and assume to be zero-centered. Now we may (implicitly) estimate the mean $\mu = \frac{1}{p}\sum_{i=1}^{p}\phi(\boldsymbol{x}_i)$ and covariance matrix Σ over the p samples. Define a kernel matrix K over the p sampled points, and let the eigendecomposition of K be $K = U\Theta U^T$. If the eigendecomposition of Σ is $V\Lambda V^T$, then $\Sigma^{-1/2} = V\Lambda^{-1/2}V^T$. Therefore, we can rewrite the hash function as follows:

$$h(\phi(\boldsymbol{x})) = \text{sign}(\phi(\boldsymbol{x})^T V\Lambda^{-1/2}V^T\tilde{\boldsymbol{z}}_t). \tag{29}$$

Note that the non-zero eigenvalues of Λ are equal to the non-zero eigenvalues of Θ. Further, denote the k-th eigenvector of the covariance matrix as $\boldsymbol{v}_k$ and the k-th eigenvector of the kernel matrix as $\boldsymbol{u}_k$. According to the derivation of kernel PCA, when the data is zero-centered, we can compute the projection

$$\boldsymbol{v}_k^T\phi(\boldsymbol{x}) = \sum_{i=1}^{p}\frac{1}{\sqrt{\theta_k}}\boldsymbol{u}_k(i)\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}), \tag{30}$$

where the $\phi(\boldsymbol{x}_i)$ are the sampled p data points.

We complete the computation of $h(\phi(\boldsymbol{x}))$ by performing this projection over all k eigenvectors, resulting in the following expression:

$$\phi(\boldsymbol{x})^T V \Lambda^{-1/2} V^T \tilde{\boldsymbol{z}}_t = \sum_{k=1}^{p} \frac{1}{\sqrt{\theta_k}} \boldsymbol{v}_k^T \phi(\boldsymbol{x}) \boldsymbol{v}_k^T \tilde{\boldsymbol{z}}_t. \tag{31}$$

Substituting Eqn. 30 for each of the eigenvector inner products, we have

$$\phi(\boldsymbol{x})^T V \Lambda^{-1/2} V^T \tilde{\boldsymbol{z}}_t = \sum_{k=1}^{p} \frac{1}{\sqrt{\theta_k}} \left(\sum_{i=1}^{p} \frac{1}{\sqrt{\theta_k}} \boldsymbol{u}_k(i) \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}) \right) \left(\sum_{i=1}^{p} \frac{1}{\sqrt{\theta_k}} \boldsymbol{u}_k(i) \phi(\boldsymbol{x}_i)^T \tilde{\boldsymbol{z}}_t \right).$$

After reordering and simplifying, this yields

$$h(\phi(\boldsymbol{x})) = \begin{cases} 1, & \text{if } \sum_{i=1}^{p} \boldsymbol{w}(i) (\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x})) \geq 0 \\ 0, & \text{otherwise} \end{cases}, \tag{32}$$

where $\boldsymbol{w}(i) = \sum_{j=1}^{p} \sum_{k=1}^{p} \frac{1}{\theta_k^{3/2}} \boldsymbol{u}_k(i) \boldsymbol{u}_k(j) \phi(\boldsymbol{x}_j)^T \tilde{\boldsymbol{z}}_t$. See [39] for intermediate steps.

Hence, the desired Gaussian random vector can be expressed as $\boldsymbol{r} = \sum_{i=1}^{p} \boldsymbol{w}(i) \phi(\boldsymbol{x}_i)$, that is, a weighted sum over the feature vectors chosen from the set of p sampled database items.[8] Then, given any novel input, the hash bit is assigned by computing kernel values between the input and those sampled items.

Summary of Algorithm. Recapping, the kernelized locality-sensitive hashing algorithm consists of the following steps:

- Select p data instances and form a kernel matrix K over this data.
- Center the kernel matrix.
- Form the hash table over the $n \gg p$ database items: for each hash function $h_j(\phi(\boldsymbol{x}))$, select t indices at random from $[1, \ldots, p]$ to sample the implicit vector $\tilde{\boldsymbol{z}}_t$, and use it to assign the next hash bit for each database instance $\boldsymbol{x}$ according to $h_j(\phi(\boldsymbol{x})) = \text{sign}(\sum_i \boldsymbol{w}(i) \kappa(\boldsymbol{x}, \boldsymbol{x}_i))$.
- For each query, form its hash key using these same hash functions (same samples of p and t indices) and employ existing LSH methods to find the approximate nearest neighbors.

Matlab code for computing KLSH functions is available from the authors' websites [39].

Computational Complexity. The most expensive step in KLSH is in the single offline computation of the kernel matrix square root, which takes time $O(p^3)$. Once this matrix has been computed, each individual hash function requires $O(p^2)$ kernel function evaluations to compute its corresponding $\boldsymbol{w}$ vector (also done offline). Once $\boldsymbol{w}$ has been computed for a given hash function, the computation of the hash

[8] Note that the random vector $\boldsymbol{r}$ constructed during the KLSH routine is only *approximately* distributed according to $\mathcal{N}(0, I)$—the central limit theorem assumes that the mean and covariance of the data are known exactly, whereas KLSH employs an approximation using a sample of p points.

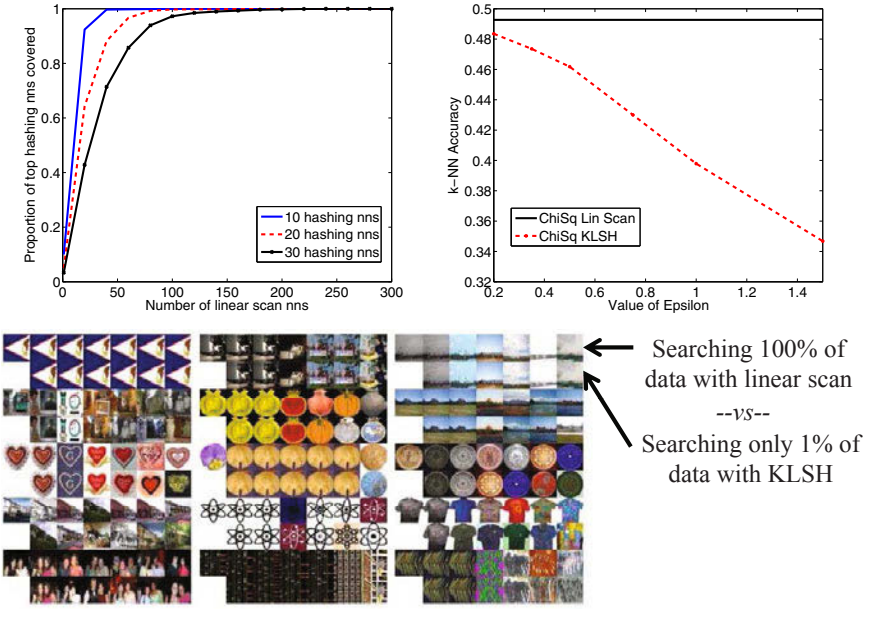


Fig. 16 Results using KLSH [39] to search the 80 Million Tiny Images data set (top left) and Flickr scenes dataset (top right) with useful image kernels—a Gaussian RBF learned kernel on Gist, and the χ^2-kernel on local features, respectively. Top left: Plot shows how many linear scan neighbors are needed to cover the first 10, 20, or 30 KLSH hashing neighbors. The ideal curve would reach the top left corner of the plot. Top right: Plot shows k-nearest neighbor accuracy of a linear scan and the KLSH algorithm as a function of LSH's ε parameter, revealing how hashing accuracy approaches that of a linear scan for smaller values of ε. Bottom: Example Tiny Image queries and the retrieved result using either a linear scan or KLSH.

function can be computed with p evaluations of the kernel function. In order to maintain efficiency, we want p to be much smaller than n—for example, $p = \sqrt{n}$ would guarantee that the algorithm maintains sub-linear search times. Empirical results for various large-scale image search tasks done in [39] suggest relatively few samples are sufficient to compute a satisfactory random vector (e.g., $p = 300$ and $t = 30$, for n up to 80 million).

Illustrative Results. Figure 16 shows some example results using KLSH for image search. In both cases, kernels are employed that would not be supported by any previous LSH algorithm. The example image retrievals show qualitatively that KLSH often retrieves neighbors very similar to those of a linear scan, but does so by searching less than 1% of the 80 Million images. At the same time, the quantitative results show exactly how much accuracy is traded off. The 10-hashing NN's curve on the Tiny Images data (top left) shows, for example, that 100% of the neighbors in KLSH's top ten are within the top 50 returned with an exhaustive linear scan.

4.4 Other Unsupervised Methods

A few other methods in the vision and learning literature tackle the problem of unsupervised binary embeddings for different metrics. Most related to some of the techniques here, Athitsos *et al.* [2, 3] propose a boosting-based approach which gives a parametric function for mapping points to binary vectors, and can accommodate metric and non-metric target similarity functions. Salakhutdinov and Hinton [56] use a neural network trained with an NCA objective [26] to build codes for text-documents. Both these approaches are explored in Torralba *et al.* [65], as detailed in Section 3.3.1 and Section 3.3.2, but with the similarity function being defined by Euclidean distance rather than label overlap. Most recently, Kulis and Darrell [38] use a kernel-based approach that jointly learns a set of projections that minimize reconstruction error. This objective can be directly and efficiently minimized using coordinate-descent.

5 Conclusions

We have reviewed a variety of methods for learning compact and informative binary projections for image data. Some are purely unsupervised (e. g. Spectral Hashing), but most can be applied in both supervised and unsupervised settings. As illustrated by the results displayed in this chapter, they offer crucial scalability for useful image search problems.

Despite their common goal, the approaches draw on a wide range techniques, including random projections, spectral methods, neural networks, boosting, and kernel methods. This diversity reflects the open nature of the problem and the extensive attention it has received lately. We anticipate that advances in machine learning and algorithms will continue to be relevant to this problem of great practical interest.

Acknowledgements. KG and RF would like to thank their co-authors Prateek Jain, Brian Kulis, Antonio Torralba and Yair Weiss for their contributions to the work covered in the chapter. They also thank the IEEE for permitting the reproduction of figures from the authors' CVPR/ICCV/PAMI papers. Finally, thanks to the Flickr users kevgibbo, bridgepix, RickC, ell brown, pdbreen, and watchsmart for sharing their photos appearing in the location recognition example of Figure 1 under the Creative Commons license.

References

1. Andoni, A., Indyk, P.: Near-Optimal Hashing Algorithms for Near Neighbor Problem in High Dimensions. In: IEEE Symposium on Foundations of Computer Science, FOCS (2006)
2. Athitsos, V., Alon, J., Sclaroff, S., Kollios, G.: BoostMap: A Method for Efficient Approximate Similarity Rankings. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2004)

3. Athitsos, V., Alon, J., Sclaroff, S., Kollios, G.: BoostMap: An Embedding Method for Efficient Nearest Neighbor Retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 30(1) (2008)
4. Babenko, B., Branson, S., Belongie, S.: Similarity Metrics for Categorization: from Monolithic to Category Specific. In: Proceedings of the IEEE International Conference on Computer Vision, ICCV (2009)
5. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning a Mahalanobis Metric from Equivalence Constraints. Journal of Machine Learning Research 6, 937–965 (2005)
6. Belkin, M., Niyogi, P.: Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In: Neural Information Processing Systems (NIPS), pp. 585–591 (2001)
7. Belkin, M., Niyogi, P.: Towards a theoretical foundation for laplacian based manifold methods. J. of Computer System Sciences (2007)
8. Bengio, Y., Paiement, J.-F., Vincent, P., Delalleau, O., Le Roux, N., Ouimet, M.: Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. In: Neural Information Processing Systems, NIPS (2004)
9. Bentley, J.: Multidimensional Divide and Conquer. Communications of the ACM 23(4), 214–229 (1980)
10. Broder, A.: On the Resemblance and Containment of Documents. In: Proceedings of the Compression and Complexity of Sequences (1997)
11. Bronstein, M., Bronstein, A., Michel, F., Paragios, N.: Data Fusion through Cross-modality Metric Learning using Similarity-Sensitive Hashing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2010)
12. Charikar, M.: Similarity Estimation Techniques from Rounding Algorithms. In: ACM Symp. on Theory of Computing (2002)
13. Chum, O., Perdoch, M., Matas, J.: Geometric min-Hashing: Finding a (Thick) Needle in a Haystack. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2009)
14. Chum, O., Philbin, J., Zisserman, A.: Near Duplicate Image Detection: min-Hash and tf-idf Weighting. In: British Machine Vision Conference (2008)
15. Coifman, R., Lafon, S., Lee, A.B., Maggioni, M., Nadler, B., Warner, F., Zucker, S.W.: Geometric Diffusions as a Tool for Harmonic Analysis and Struture Definition of Data: Diffiusion Maps. Proc. Natl. Academy of Sciences 102(21), 7426–7431 (2005)
16. Crammer, K., Keshet, J., Singer, Y.: Kernel Design Using Boosting. In: Neural Information Processing Systems, NIPS (2002)
17. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.: Locality-Sensitive Hashing Scheme Based on p-Stable Distributions. In: Symposium on Computational Geometry, SOCG (2004)
18. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image Retrieval: Ideas, Influences, and Trends of the New Age. ACM Computing Surveys (2008)
19. Davis, J., Kulis, B., Jain, P., Sra, S., Dhillon, I.: Information-Theoretic Metric Learning. In: Proceedings of International Conference on Machine Learning, ICML (2007)
20. Fergus, R., Bernal, H., Weiss, Y., Torralba, A.: Semantic Label Sharing for Learning with Many Categories. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6311, pp. 762–775. Springer, Heidelberg (2010)
21. Fowlkes, C., Belongie, S., Chung, F., Malik, J.: Spectral Grouping Using the Nystrom Method. PAMI 26(2), 214–225 (2004)
22. Freidman, J., Bentley, J., Finkel, A.: An Algorithm for Finding Best Matches in Logarithmic Expected Time. ACM Transactions on Mathematical Software 3(3), 209–226 (1977)
23. Gionis, A., Indyk, P., Motwani, R.: Similarity Search in High Dimensions via Hashing. In: Proc. Intl Conf. on Very Large Data Bases (1999)
24. Globerson, A., Roweis, S.: Metric Learning by Collapsing Classes. In: Neural Information Processing Systems, NIPS (2005)

25. Goemans, M., Williamson, D.: Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. JACM 42(6), 1115–1145 (1995)
26. Goldberger, J., Roweis, S.T., Salakhutdinov, R.R., Hinton, G.E.: Neighborhood Components Analysis. In: Neural Information Processing Systems, NIPS (2004)
27. Grauman, K., Darrell, T.: The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In: Proceedings of the IEEE International Conference on Computer Vision, ICCV (2005)
28. Grauman, K., Darrell, T.: Pyramid Match Hashing: Sub-Linear Time Indexing Over Partial Correspondences. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2007)
29. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality Reduction by Learning an Invariant Mapping. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2006)
30. Hertz, T., Bar-Hillel, A., Weinshall, D.: Learning Distance Functions for Image Retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2004)
31. Hertz, T., Bar-Hillel, A., Weinshall, D.: Learning a Kernel Function for Classification with Small Training Samples. In: Proceedings of International Conference on Machine Learning, ICML (2006)
32. Hinton, G.E., Salakhutdinov, R.R.: Reducing the Dimensionality of Data with Neural Networks. Nature 313(5786), 504–507 (2006)
33. Indyk, P., Motwani, R.: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In: 30th Symposium on Theory of Computing (1998)
34. Indyk, P., Thaper, N.: Fast Image Retrieval via Embeddings. In: Intl. Workshop on Statistical and Computational Theories of Vision (2003)
35. Iqbal, Q., Aggarwal, J.K.: CIRES: A System for Content-Based Retrieval in Digital Image Libraries. In: International Conference on Control, Automation, Robotics and Vision (2002)
36. Jain, P., Kulis, B., Dhillon, I., Grauman, K.: Online Metric Learning and Fast Similarity Search. In: Neural Information Processing Systems, NIPS (2008)
37. Jain, P., Kulis, B., Grauman, K.: Fast Image Search for Learned Metrics. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2008)
38. Kulis, B., Darrell, T.: Learning to Hash with Binary Reconstructive Embeddings. In: Neural Information Processing Systems, NIPS (2009)
39. Kulis, B., Grauman, K.: Kernelized Locality-Sensitive Hashing for Scalable Image Search. In: Proceedings of the IEEE International Conference on Computer Vision, ICCV (2009)
40. Kulis, B., Jain, P., Grauman, K.: Fast Similarity Search for Learned Metrics. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 31 (2009)
41. Lazebnik, S., Schmid, C., Ponce, J.: Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2006)
42. Lin, R.-S., Ross, D., Yagnik, J.: SPEC Hashing: Similarity Preserving Algorithm for Entropy-based Coding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2010)
43. Ling, H., Soatto, S.: Proximity Distribution Kernels for Geometric Context in Category Recognition. In: Proceedings of the IEEE International Conference on Computer Vision, ICCV (2007)
44. Liu, T., Moore, A., Gray, A., Yang, K.: An Investigation of Practical Approximate Nearest Neighbor Algorithms. In: Neural Information Processing Systems, NIPS (2005)
45. Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision (IJCV) 60(2) (2004)
46. Mu, Y., Shen, J., Yan, S.: Weakly-supervised hashing in kernel space. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2010)

47. Muja, M., Lowe, D.: Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In: International Conference on Computer Vision Theory and Application, VISSAPP (2009)
48. Nadler, B., Lafon, S., Coifman, R., Kevrekidis, I.: Diffusion maps, spectral clustering and reaction coordinates of dynamical systems (2008), http://arxiv.org
49. Ng, A., Jordan, M.I., Weiss, Y.: On Spectral Clustering, Analysis and an Algorithm. In: Neural Information Processing Systems, NIPS (2001)
50. Oliva, A., Torralba, A.: Modeling the Shape of the Scene: a Holistic Representation of the Spatial Envelope. International Journal in Computer Vision 42, 145–175 (2001)
51. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object Retrieval with Large Vocabularies and Fast Spatial Matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2007)
52. Raginsky, M., Lazebnik, S.: Locality-Sensitive Binary Codes from Shift-Invariant Kernels. In: Neural Information Processing Systems, NIPS (2009)
53. Rahimi, A., Recht, B.: Random Features for Large-Scale Kernel Machines. In: Neural Information Processing Systems, NIPS (2007)
54. Rice, J.: Mathematical Statistics and Data Aanalysis. Duxbury Press (2001)
55. Roweis, S., Saul, L.: Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science 290(5500), 2323–2326 (2000)
56. Salakhutdinov, R.R., Hinton, G.E.: Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure. In: AISTATS (2007)
57. Salakhutdinov, R.R., Hinton, G.E.: Semantic Hashing. In: SIGIR Workshop on Information Retrieval and Applications of Graphical Models (2007)
58. Schölkopf, B., Smola, A., Müller, K.-R.: Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation 10, 1299–1319 (1998)
59. Schultz, M., Joachims, T.: Learning a Distance Metric from Relative Comparisons. In: Neural Information Processing Systems, NIPS (2003)
60. Shakhnarovich, G.: Learning Task-Specific Similarity. PhD thesis. MIT (2005)
61. Shakhnarovich, G., Viola, P., Darrell, T.: Fast pose estimation with parameter sensitive hashing. In: Proceedings of the IEEE International Conference on Computer Vision, ICCV (2003)
62. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
63. Sivic, J., Zisserman, A.: Video Google: A Text Retrieval Approach to Object Matching in Videos. In: Proceedings of the IEEE International Conference on Computer Vision, ICCV (2003)
64. Tenenbaum, J., de Silva, V., Langford, J.: A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science 290(5500), 2319–2323 (2000)
65. Torralba, A., Fergus, R., Weiss, Y.: Small Codes and Large Image Databases for Recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2008)
66. Uhlmann, J.: Satisfying General Proximity/Similarity Queries with Metric Trees. Information Processing Letters 40, 175–179 (1991)
67. van der Maaten, L., Hinton, G.: Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9, 2579–2605 (2008)
68. Varma, M., Ray, D.: Learning the Discriminative Power-Invariance Trade-off. In: Proceedings of the IEEE International Conference on Computer Vision, ICCV (2007)
69. Wang, J., Kumar, S., Chang, S.-F.: Semi-Supervised Hashing for Scalable Image Retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2010)
70. Wang, J., Kumar, S., Chang, S.-F.: Sequential Projection Learning for Hashing with Compact Codes. In: Proceedings of International Conference on Machine Learning, ICML (2010)
71. Weinberger, K., Blitzer, J., Saul, L.: Distance Metric Learning for Large Margin Nearest Neighbor Classification. In: Neural Information Processing Systems, NIPS (2006)

72. Weiss, Y., Torralba, A., Fergus, R.: Spectral Hashing. In: Neural Information Processing Systems, NIPS (2008)
73. Xing, E., Ng, A., Jordan, M., Russell, S.: Distance Metric Learning, with Application to Clustering with Side-Information. In: Neural Information Processing Systems, NIPS (2002)
74. Xu, D., Cham, T.J., Yan, S., Chang, S.-F.: Near Duplicate Image Identification with Spatially Aligned Pyramid Matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2008)
75. Yeh, T., Grauman, K., Tollmar, K., Darrell, T.: A Picture is Worth a Thousand Keywords: Image-Based Object Search on a Mobile Platform. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (2005)
76. Zhang, H., Berg, A., Maire, M., Malik, J.: SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2006)
77. Zhang, J., Marszalek, M., Lazebnik, S., Schmid, C.: Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. International Journal of Computer Vision (IJCV) 73(2), 213–238 (2007)

Bayesian Painting by Numbers: Flexible Priors for Colour-Invariant Object Recognition

Jeroen C. Chua, Inmar E. Givoni, Ryan P. Adams, and Brendan J. Frey

Abstract. Generative models of images should take into account transformations of geometry and reflectance. Then, they can provide explanations of images that are factorized into intrinsic properties that are useful for subsequent tasks, such as object classification. It was previously shown how images and objects within images could be described as compositions of regions called structural elements or 'stels'. In this way, transformations of the reflectance and illumination of object parts could be accounted for using a hidden variable that is used to 'paint' the same stel differently in different images. For example, the stel corresponding to the petals of a flower can be red in one image and yellow in another. Previous stel models have used a fixed number of stels per image and per image class. Here, we introduce a Bayesian stel model, the *colour-invariant admixture* (CIA) model, which can infer different numbers of stels for different object types, as appropriate. Results on Caltech101 images show that this method is capable of automatically selecting a number of stels that reflects the complexity of the object class and that these stels are useful for object recognition.

1 Introduction

Vision can be thought of as inference in a learnt model of the relationships between spatial patterns at different levels of abstraction. Marr [20] described three levels of visual patterns: the primal sketch, corresponding to what an artist would draw to represent parts of objects in a scene; the 2.5D sketch, which overlays the primal sketch with textures, colours and shading; and the 3D model, which relates primal and 2.5D sketches derived from different viewpoints and 3D manipulations. Two extreme approaches to developing visual learning algorithms include using highly

Jeroen C. Chua · Inmar E. Givoni · Ryan P. Adams · Brendan J. Frey
University of Toronto, Toronto, Canada
e-mail: {jeroen,inmar,radams,frey}@psi.toronto.edu

R. Cipolla et al. (Eds.): Machine Learning for Computer Vision, SCI 411, pp. 89–117.
springerlink.com

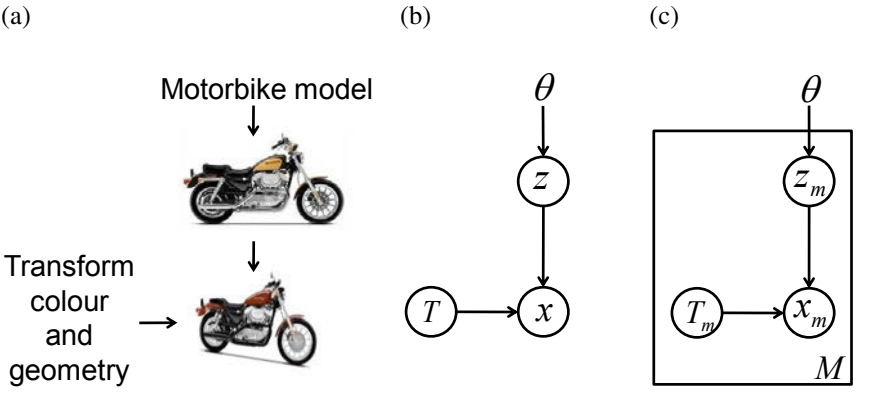


Fig. 1 Factorizing image explanations into intrinsic transformations. (a) A model is used to generate a description of the appearance of a motorbike, which is then modified by transformations of geometry and reflectance to produce an instance. (b) This process can be formalized by thinking of the observed image x as a random variable. It is assumed to have been generated by applying an image-specific transformation T to a latent image z so as to change various object properties, including reflectance and geometry. The parameter θ describes the distribution over normalized latent images, z. (c) A dataset of M images $x_1, \ldots, x_M$ is generated using corresponding latent images and transformations. Here, plate notation is used, where variables within the box are replicated M times corresponding to the M images.

flexible, unstructured neural networks [10, 11, 27, 5], and using highly structured techniques that hard-wire sensible rules for pattern generation [9, 21, 31]. In the case of neural network approaches, the hope is that Marr's different levels of patterns will emerge after learning in a deep neural network, because they are the most efficient way to model the statistics of images [10, 11]. In the case of methods using hard-wired pattern rules, the hope is that a reasonably simple set of rules can be combined with a straightforward inference algorithm to accurately describe the huge variation seen in natural images [21].

We take an approach that combines the best aspects of the neural network and pattern rule approaches, by exploring highly flexible statistical models that incorporate sensible pattern rules. The best known example of this approach is the convolutional neural network [17], which takes an image as input, propagates signals through multiple layers of hidden variables, and then predicts the class of the object in the image. The layers of variables are arranged according to the topology of the input image, and each hidden variable receives input only from nearby variables in the previous layer. This method achieves state-of-the-art performance on several standard classification tasks [17]. In our approach, we recognize that in general the number of labels that are available for training is exponentially smaller than the number of possible pattern combinations. Therefore, we use statistical models of the image data itself and train these models in an unsupervised fashion.

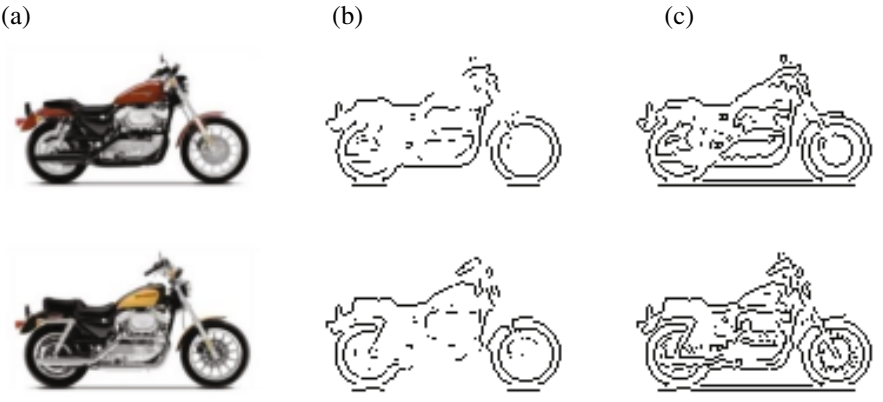


Fig. 2 Attempting to detect object parts using image gradients. Two Caltech101 images (a) were processed using a Canny edge detector with stringent (b) and liberal (c) thresholds. Object parts are either not delineated or are delineated but accompanied by spurious irrelevant edges.

Fig. 1 illustrates our approach, which was first described in [6, 13, 7, 14]. Variations due to object location, orientation, scale, reflectance and illumination are factored out and represented in a transformation variable T, and unsupervised learning methods are used to model the normalized, latent image z. From a generative point of view, each image in the dataset is assumed to have been produced by generating a latent image from the model $p(z|\theta)$, randomly selecting a transformation T from $p(T)$, and then applying the transformation to obtain the observed image according to the rendering model $p(x|T,z)$. When an object is most naturally described as a composition of articulating, deformable parts, the transformation T should be factorized into a field of transformations where each sub-transformation transforms an object part.

Here we attend to the vision problem of accounting for variability in reflectance properties and illumination across object instances, so we will assume that all instances of an object have similar geometry. Fig. 2(a) shows two motorbike images from the Caltech101 dataset [18]. The motorbikes and the parts comprising them have similar geometry, but quite different reflectance and illumination properties. For example, a prominent difference between the two images is the colour of the pipework; whereas the first motorbike has black pipework, the second motorbike has light chrome pipework.

A popular standard approach to reducing sensitivity to variations in reflectance and illumination is to pre-whiten images so as to emphasize edges [23]. Absolute pixel intensities are discarded and instead only information about edges [2] or oriented intensity gradients, such as those encoded by SIFT features [19], are used. This approach produced state-of-the-art results on image classification problems in the first decade of this century [16]. However, it is sensitive to parameters such as the edge detector sensitivity, the patch size used to define SIFT features, thresholds

on minimum gradients, the degree of contrast normalization, and so on. Figs. 2(b) and 2(c) illustrate the difficulty in selecting thresholds for a Canny edge detector [2]. The more stringent threshold used in Fig. 2(b) leads to important edges being lost in regions of low contrast, such as the gas tank in the top image and the pipework in the bottom image. The more liberal threshold used in Fig. 2(c) leads to a large number of spurious edges. The essential problem here is that it is not possible to define beforehand an edge detection threshold that will differentiate all object parts without also erroneously detecting edges due to noise.

A quite different approach to building robustness against variation in reflectance and illumination was proposed in [15, 14]. They defined a 'structure element' (stel) as a class-specific region in the image plane that can have different texture, shading and colour in different examples, but whose spatial structure is similar across images. Stels correspond to regions in Marr's primal sketch, which can be rendered differently in different 2.5D sketches. Stels are identified by indices, so that the latent image z in Fig. 1 is an image of stel indices. The latent image model $p(z|\theta)$ provides a distribution over index maps. For the current image, T is a colour model that specifies a distribution over colours for each stel index. Given the current index map and colour model, a distribution over colours is specified for every pixel.

Stel models account for appearance in a way that factorizes out instance-specific reflectance and illumination properties. Given a training set of images, the learnt stel model segments images from an object class into different regions (stels) in a colour-invariant way by modelling the co-occurrence of colours within an image and spatial relationships across images within the object class. Pixels that are typically the same colour and can be grouped into similar shapes across images are put into a single stel, which loosely corresponds to an object part. Grouping pixels in this way provides a class-specific bias for parts-based segmentation of training and test images. In the context of object recognition, stel models provide a means to model spatial relationships between oriented gradient features [15, 24].

Using the expectation-maximization (EM) algorithm described in [15], a single-class model with nine stels was learnt from the five Caltech101 translation- and scale-normalized images shown in Fig. 3(a). The image sizes were 75×132 and they were converted to greyscale for analysis. Fig. 3(b) shows the nine stels, where for each stel an image of probabilities that pixels belong to the stel are shown, with white corresponding to a probability of one and black corresponding to a probability of zero. Some stels, such as the stel in the middle row on the left, account for large portions of pixels. Other stels, such as the one in the middle that accounts for the front wheel disk, account for small portions of pixels. The three dominant stels are shown in the left column.

Since stels are defined in a way that is similar to Marr's definition of the primal sketch, an interesting question is whether the learnt stels can be used as a primal sketch. Recall that the problem with the image-derived edge maps shown in Figs. 2(b) and 2(c) is that it is not possible to pick a threshold that yields an edge map that clearly delineates objects and parts, while at the same time not introducing many erroneous edges. Since stels are required to be consistent across

(a)

(b)

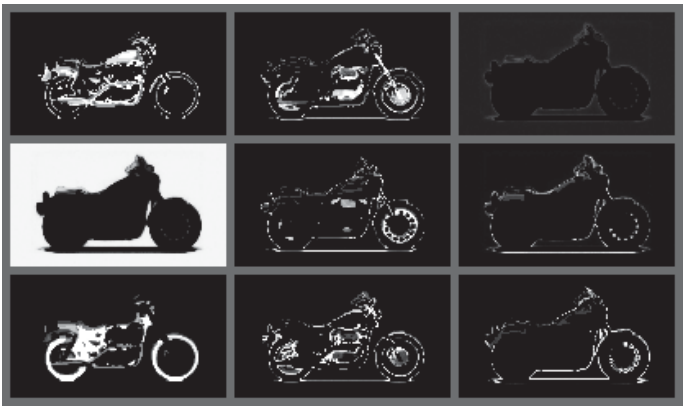


Fig. 3 The structure elements (stels) of motorbike images. Five images from the Caltech101 database (a) were used to learn nine stels (b). For each stel, an image of probabilities that pixels belong to the stel is shown. The first stel accounts for the pipework, gas tank and rear fender.

Fig. 4 Primal sketches derived from stels. These three edge maps were obtained from smoothed versions of the three dominant stels shown in the left column of Fig. 3(b). They correspond to 'primal sketches' of the motorbike outline, the wheels and seat, and the pipework and gas tank.

images, can they be used to make primal sketches that account for object parts? To answer this question, the three dominant stels were smoothed using a Gaussian filter with $\sigma = 1.5$ and the MATLAB Canny edge detector was applied using default settings. The resulting edge maps are shown in Fig. 4 and correspond to the overall outline of the motorbike, the wheels and seat, and the pipework and gas tank.

Since stel segmentations capture interesting spatial relationships within an object class, they can be used in an object recognition framework by using the segmentations to encode the spatial configuration of local image features. It has been previously shown that recognition performance can be improved by incorporation of the spatial relationships between features [24], in contrast to models based on bags of visual words derived from, e.g., SIFT features.

A major drawback of current approaches to modelling with stels is that they require the number of stels — the number of object parts for an object class — to be fixed in advance. As it is difficult to determine a class-appropriate number of stels *a priori*, this is an undesirable requirement. Using too few stels may result in segmentations that are too coarse, while using an excessive number of stels may lead to overfitting. Furthermore, differing poses and lighting conditions may call for a different number of stels even within a single object class. This important free parameter has typically been set by hand or by using computationally-expensive cross-validation.

Here, we propose a Bayesian stel model that uses a prior distribution over the assignment of pixels to stels to regularize the complexity of the stel segmentation. After learning, the posterior distribution captures information about the appropriate distribution over stels for a given set of data. We develop a framework for stels that models images as an admixture, complementing other approaches, such as latent Dirichlet allocation [1, 29, 28, 3].

2 The Colour-Invariant Admixture Model

One powerful approach to modelling data is to use an admixture, which captures the idea that a given datum (e.g., an image) may be a combination of several latent components. This idea has found wide use in the modelling of natural language documents, where latent Dirichlet allocation (LDA) [1] provides a particularly convenient and elegant generative probabilistic model for exchangeable text data. When considering the problem of vision from a modelling point of view, Marr's notion of a primal sketch maps well onto the admixture concept. Considering again the stel-derived edge maps in Fig. 4, we can imagine that these sketches are blended together to produce the observed image. Note that this is in contrast to a simple mixture model, where images would result from precisely one of these three sketches.

In this section we develop a generative Bayesian variant of the stel model, which we call the *colour-invariant admixture* (CIA) model. This model extends the standard approach to stel modelling to enable representation of the full posterior distribution over the stels. By combining the powerful ability to learn spatial relationships using stels, with the flexible invariance properties of a fully-probabilistic latent colour model, CIA is able to learn image-specific properties of colour that enable richer feature extraction for supervised learning tasks. Inference is straightforward, using Markov chain Monte Carlo.

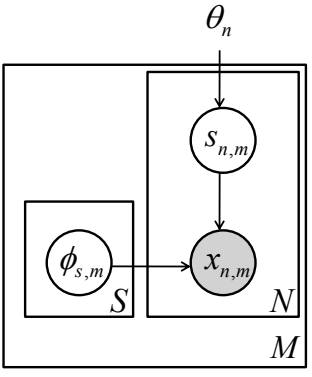


Fig. 5 Graphical model for the standard stel model

2.1 The Standard Stel Model

We first formally describe the standard stel model, as outlined in [15]. We assume that there are M images, each with N pixels. We denote the nth pixel in the mth image by $x_{n,m}$, taking values in a colour space $\mathcal{C}$, which we will take without loss of generality to be $\mathbb{R}^D$. We denote the collection of all pixel values across all images by $\mathbf{X} = \{\{x_{n,m}\}_{n=1}^{N}\}_{m=1}^{M}$.

The standard stel approach associates with each pixel measurement $x_{n,m}$ a discrete variable $s_{n,m} \in \{1,2,\ldots,S\}$, which indicates the stel to which that pixel belongs. We denote this set of indices as $\Xi = \{\{s_{n,m}\}_{n=1}^{N}\}_{m=1}^{M}$. A stel can be loosely thought of as either a background model, or an object part. For instance, for the motorcycle class, one stel may represent the wheels, another stel may represent the pipework, and another stel may represent the background.

The main assumption of the stel model is that pixels belonging to the same stel have high probability under a tight distribution defined on $\mathcal{C}$. That is, the stel identity of a pixel is highly informative about colour. The key insight is to allow these distributions to vary across images, i.e., in image m, stel s has unique parameters $\phi_{s,m}$. If the observation model $p(x \mid \phi)$ is a Gaussian distribution on $\mathcal{C}$, for example, then $\phi_{s,m}$ would be mean and covariance parameters that are specific to the combination of s and m. We denote the aggregate set of colour-distribution parameters for the training images as $\Phi = \{\{\phi_{s,m}\}_{s=1}^{S}\}_{m=1}^{M}$.

The statistical sharing across images occurs through the per-pixel multinomial distribution over the stel assignments, parameterized by $\boldsymbol{\theta}_n$, where $\theta_{s,n} \geq 0$ and $\sum_s \theta_{s,n} = 1$. We denote the aggregate set of index distributions as $\Theta = \{\boldsymbol{\theta}_n\}_{n=1}^{N}$. The standard stel graphical model is shown in Fig. 5.

Note that since each image can have unique colour distributions, the same object part can have different colours in different images. Therefore, the inferred assignments of stels will be invariant to colour in the sense that what matters is not the *specific* colour, but colour *co-occurrence*. It is not necessary for object parts to be

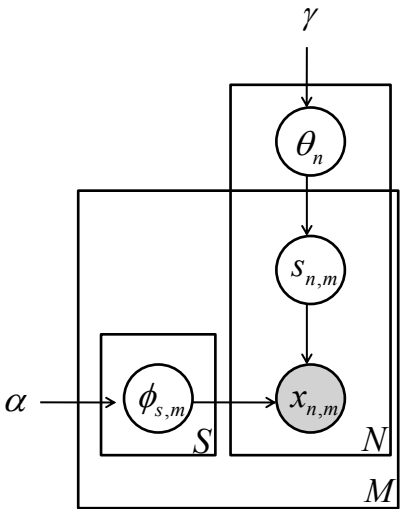


Fig. 6 Graphical model for CIA, where *s* indexes stels, *n* indexes pixels and *m* indexes images.

of the same colour in all training images; all that is required is that object parts regularly co-occur in colour across images. Note, however, that the training set must be normalized in position, orientation, and scale, since inference of stel assignments is on a per-pixel basis, and the probability distributions $\boldsymbol{\theta}_n$ are shared across all images. It is possible to add in deformation variables to deal with, e.g., translation and rotation. However, even without such deformation variables, the stel model is capable of handling small amounts of deformation via soft stel assignments [24].

Learning of the parameters Θ and Φ can be performed via maximum likelihood using the expectation maximization (EM) algorithm as in [15]. As the pixel-wise stel distributions $\boldsymbol{\theta}_n$ are shared across the entire training set and are invariant to colour, the stel model can provide a robust prior distribution over segmentation for a given object class. Given a test image, the posterior stel segmentation can be efficiently inferred, and this segmentation can be used for other tasks such as object recognition.

2.2 The Stel Model as a Generative Admixture

In practice, the performance of the stel model is very sensitive to the regularization on Θ. Using too strong of a regularization results in coarse, uninformative image segmentations, but allowing too much flexibility results in overfitting. To handle this difficulty, we propose a Bayesian approach that is capable of maintaining a full posterior distribution over Θ. This helps to relieve overfitting, but still results in a flexible model. At test time, the uncertainty in the stel parameters can be taken into

account when performing segmentation and object recognition. Additionally, this approach enables CIA to be used as a module in larger hierarchical models with little modification.

As before, $s_{n,m} \in \{1,2,\ldots,S\}$ is the stel assignment of pixel n in image m, and $\boldsymbol{\theta}_n$ specifies the multinomial distribution over stel indices for pixel n. We will assume that the pixel observation model is a Gaussian distribution with unknown mean and covariance, i.e., $\phi_{s,m} = \{\mu_{s,m}, \Lambda_{s,m}\}$. We place a Dirichlet prior on the $\boldsymbol{\theta}_n$ with base measure γ. We place normal-inverse-Wishart (NIW) priors on the $\phi_{s,m}$, with parameters $\alpha = \{\mu_0, \kappa_0, \nu_0, \Lambda_0\}$. This leads to the following generative model:

$$\boldsymbol{\theta}_n \,|\, \gamma \sim \text{Dirichlet}(\gamma) \tag{1}$$

$$s_{n,m} \,|\, \boldsymbol{\theta}_n \sim \text{Multinomial}(\boldsymbol{\theta}_n, 1) \tag{2}$$

$$\phi_{s,m} \,|\, \alpha \sim \text{NIW}(\mu_0, \kappa_0, \nu_0, \Lambda_0) \tag{3}$$

$$x_{n,m} \,|\, s_{n,m}, \{\phi_{s,m}\}_{s=1}^{S} \sim \text{Norm}(\mu_{s_{n,m},m}, \Lambda_{s_{n,m},m}). \tag{4}$$

The graphical model for this generative procedure is provided in Fig. 6. In this paper we will use parameterization of the normal-inverse-Wishart given by [22]:

$$\text{NIW}(\mu, \Sigma \,|\, \mu_0, \kappa_0, \nu_0, \Lambda_0) = \frac{|\Lambda_0|^{\nu_0/2}}{2^{D\nu_0/2}\Gamma_D(\nu_0/2)(2\pi/\kappa_0)^{D/2}} |\Sigma|^{-(\nu_0+D)/2-1} \times \exp\left\{-\frac{1}{2}\text{tr}\left(\Lambda_0\Sigma^{-1}\right) - \frac{\kappa_0}{2}(\mu-\mu_0)^{\mathsf{T}}\Sigma^{-1}(\mu-\mu_0)\right\}, \tag{5}$$

where $\Gamma_D(\cdot)$ is the generalized gamma function given by

$$\Gamma_D(z) = \pi^{D(D-1)/4} \prod_{d=1}^{D} \Gamma\left(\frac{1}{2}(2z+1-D)\right). \tag{6}$$

In this generative model, we are specifying a full joint distribution over the training images that possesses two different kinds of sharing across the data. In the first case, all of the images (which are assumed to be from a single object class), share the parameter $\boldsymbol{\theta}_n$ that provides the distribution over stels at the pixel level. However, each individual image can use different actual stel assignments to account for variations in object boundaries, deformations, etc. Since the pixels within an image are conditionally independent given Θ and Φ, multiple stels can be represented in a single image. This is the heart of the admixture idea. The second type of sharing is within a single image: each image has its own unique set of distributions associated with its colours. This enables robustness to variation in reflectance and illumination, but supports the intuitive inductive bias that all the pixels for a single stel within an image should have similar properties.

2.3 Inference via Gibbs Sampling

Having specified a stel-based generative model, we can now examine the task of learning, which in this case corresponds to finding the marginal posterior over the stel distributions, given the data and hyperparameters:

$$p(\Theta \,|\, \mathbf{X}, \gamma, \alpha) = \int \mathrm{d}\Phi \int \mathrm{d}\Xi \, p(\Theta, \Phi, \Xi \,|\, \mathbf{X}, \gamma, \alpha). \tag{7}$$

The integrals required to compute this marginal distribution are intractable, however. We therefore take an approximate inference approach based on numerical integration via Markov chain Monte Carlo (MCMC). We begin by taking the generative model specified in Eqs.(1)-(4) and constructing a joint distribution over the data and the unknowns, given the hyperparameters γ and α:

$$p(\mathbf{X}, \Xi, \Theta, \Phi \,|\, \gamma, \alpha) \propto p(\Theta, \Phi, \Xi \,|\, \mathbf{X}, \gamma, \alpha). \tag{8}$$

This distribution is proportional to the posterior distribution over all unknowns that appears inside the integral in Eq. (7). Although we are primarily interested in Θ, by examining the graphical model in Fig. 6, we observe that the posterior distribution over Θ can be easily computed given Ξ and γ. Therefore, it is sufficient for our purposes to generate samples from the posterior distribution over Ξ, marginalizing over both Θ and Φ:

$$p(\Xi \,|\, \mathbf{X}, \gamma, \alpha) \propto p(\mathbf{X}, \Xi \,|\, \gamma, \alpha) = \int \mathrm{d}\Phi \int \mathrm{d}\Theta \, p(\mathbf{X}, \Xi, \Phi, \Theta \,|\, \gamma, \alpha). \tag{9}$$

We further observe that the distribution in Eq. (9) factorizes as

$$\begin{aligned} &= \int \mathrm{d}\Phi \int \mathrm{d}\Theta \, p(\mathbf{X} \,|\, \Xi, \Phi) \, p(\Xi \,|\, \Theta) \, p(\Theta \,|\, \gamma) \, p(\Phi \,|\, \alpha) \\ &= \left[\int \mathrm{d}\Phi \, p(\mathbf{X} \,|\, \Xi, \Phi) \, p(\Phi \,|\, \alpha) \right] \left[\int \mathrm{d}\Theta \, p(\Xi \,|\, \Theta) \, p(\Theta \,|\, \gamma) \right]. \end{aligned} \tag{10}$$

When constructing the generative stel model, we could have used various priors for $p(\Phi \,|\, \alpha)$ and $p(\Theta \,|\, \gamma)$. However, our specific choice of NIW and Dirichlet distributions, respectively, leads to analytic solutions for the two integral factors in Eq. (10).

In the first case, we have

$$\int \mathrm{d}\Phi \, p(\mathbf{X} \,|\, \Xi, \Phi) \, p(\Phi \,|\, \alpha) = p(\mathbf{X} \,|\, \Xi, \alpha), \tag{11}$$

which we recognize as the marginal likelihood of the data (the denominator of Bayes' theorem), as partitioned by the stel assignments Ξ. Our objective will be to perform a Gibbs sweep over all assignments. To do this, we can factor Eq. (11) into

$$p(\mathbf{X}\,|\,\Xi,\alpha) = p(x_{n^\star,m^\star}\,|\,\mathbf{X}/x_{n^\star,m^\star},\Xi,\alpha)\,p(\mathbf{X}/x_{n^\star,m^\star}\,|\,\Xi,\alpha). \tag{12}$$

The first factor is the posterior predictive distribution given all data except for pixel $n^\star$ in image $m^\star$. The second term is a constant which does not depend on the pixel we are currently updating. To compute the predictive distribution for the triplet of stel s, pixel n and image m, we begin by finding the sufficient statistics of the data *excluding* pixel n in image m. Let $\delta(s,s')$ be the Kronecker delta function, which takes value one if s and s' are equal and zero otherwise. The sufficient statistics are then:

$$N_{s,n,m} = \sum_{n'\neq n}^{N} \delta(s,s_{n',m}) \tag{13}$$

$$\bar{x}_{s,n,m} = \frac{1}{N_{s,n,m}} \sum_{n'\neq n}^{N} \delta(s,s_{n',m})\,x_{n',m} \tag{14}$$

$$\bar{\boldsymbol{X}}_{s,n,m} = \frac{1}{N_{s,n,m}} \sum_{n'\neq n}^{N} \delta(s,s_{n',m}) \left[(x_{n',m}-\bar{x}_{s,n,m})(x_{n',m}-\bar{x}_{s,n,m})^{\mathsf{T}}\right]. \tag{15}$$

Following the notation of [22], these statistics can be used to find the parameters of the normal-inverse-Wishart posterior on the colour distribution:

$$\kappa_{s,n,m} = \kappa_0 + N_{s,n,m} \tag{16}$$

$$\mu_{s,n,m} = \frac{\kappa_0}{\kappa_0+N_{s,n,m}}\mu_0 + \frac{N_{s,n,m}}{\kappa_0+N_{s,n,m}}\bar{x}_{s,n,m} \tag{17}$$

$$\nu_{s,n,m} = \nu_0 + N_{s,n,m} \tag{18}$$

$$\Lambda_{s,n,m} = \Lambda_0 + \bar{\boldsymbol{X}}_{s,n,m} + \frac{\kappa_0 N_{s,n,m}}{\kappa_0+N_{s,n,m}}(\bar{x}_{s,n,m}-\mu_0)(\bar{x}_{s,n,m}-\mu_0)^{\mathsf{T}}. \tag{19}$$

These parameters also provide a closed form for the posterior predictive distribution, which is a multivariate Student t-distribution:

$$p(x_{n^\star,m^\star}\,|\,\mathbf{X}/x_{n^\star,m^\star},\Xi,\alpha) = t_{\nu_{s,n,m}-D+1}\left(\mu_{s,n,m}, \frac{\Lambda_{s,n,m}(\kappa_{s,n,m}+1)}{\kappa_{s,n,m}(\nu_{s,n,m}-D+1)}\right), \tag{20}$$

where D is the dimensionality of the colour space, e.g., $D=3$ for RGB and $D=1$ for greyscale. The probability density function of the Student t-distribution is given by

$$t_\nu(x\,|\,\mu,\Sigma) = \frac{\Gamma(\nu/2+D/2)}{\Gamma(\nu/2)}\frac{|\Sigma|^{-1/2}}{(\pi\nu)^{D/2}}\left(1+\frac{1}{\nu}(x-\mu)^{\mathsf{T}}\Sigma^{-1}(x-\mu)\right)^{-(\nu+D)/2}. \tag{21}$$

In the second factor of Eq. (10), we are also computing a marginal likelihood:

$$p(\Xi \,|\, \gamma) = \int \mathrm{d}\Theta \, p(\Xi \,|\, \Theta) \, p(\Theta \,|\, \gamma)$$
$$= \int \mathrm{d}\Theta \prod_{n=1}^{N} \frac{\Gamma\left(\sum_{s=1}^{S} \gamma_s\right)}{\prod_{s=1}^{S} \Gamma(\gamma_s)} \prod_{s=1}^{S} \theta_{s,n}^{\gamma_s - 1} \prod_{m=1}^{M} \theta_{s,n}^{\delta(s, s_{n,m})}. \tag{22}$$

Introducing the sufficient statistic

$$\eta_{s,n} = \sum_{m=1}^{M} \delta(s, s_{n,m}), \tag{23}$$

we can rewrite Eq. (22) as a Dirichlet-multinomial (or Pólya) distribution:

$$p(\Xi \,|\, \gamma) = \int \mathrm{d}\Theta \prod_{n=1}^{N} \frac{\Gamma\left(\sum_{s=1}^{S} \gamma_s\right)}{\prod_{s=1}^{S} \Gamma(\gamma_s)} \prod_{s=1}^{S} \theta_{s,n}^{\gamma_s + \eta_{s,n} - 1} \tag{24}$$
$$= \prod_{n=1}^{N} \int \mathrm{d}\boldsymbol{\theta}_n \frac{\Gamma\left(\sum_{s=1}^{S} \gamma_s\right)}{\prod_{s=1}^{S} \Gamma(\gamma_s)} \prod_{s=1}^{S} \theta_{s,n}^{\gamma_s + \eta_{s,n} - 1} \tag{25}$$
$$= \prod_{n=1}^{N} \frac{\Gamma\left(\sum_{s=1}^{S} \gamma_s\right)}{\Gamma\left(\sum_{s=1}^{S} \gamma_s + \eta_{s,n}\right)} \prod_{s=1}^{S} \frac{\Gamma(\gamma_s + \eta_{s,n})}{\Gamma(\gamma_s)}. \tag{26}$$

In order to Gibbs sample, we require the conditional distribution of a single assignment $s_{n,m}$ given all the rest of Ξ. As in the NIW case, we can factorize the marginal likelihood and compute the predictive sufficient statistics for each stel-image-pixel triplet:

$$\eta_{s,n,m} = \sum_{m' \neq m}^{M} \delta(s, s_{n,m'}) = \eta_{s,n} - \delta(s, s_{n,m}). \tag{27}$$

The statistic $\eta_{s,n,m}$ is the number of images for which pixel n has been assigned to stel s, excluding image m. This leads to the posterior predictive for $s_{n,m}$ given all other assignments:

$$p(s_{n,m} = s \,|\, \Xi / s_{n,m}, \gamma) = \frac{\eta_{s,n,m} + \gamma_s}{\sum_{s'=1}^{S} \eta_{s',n,m} + \gamma_{s'}}. \tag{28}$$

The overall Gibbs sampling update is then the product of the "prior" of the assignment induced by the Dirichlet-multinomial predictive distribution in Eq. (28) and the "likelihood" of the pixel intensity that results from the Student t-distribution:

$$p(s_{n,m} = s \,|\, \Xi / s_{n,m}, \mathbf{X}, \gamma, \alpha)$$
$$\propto \frac{\eta_{s,n,m} + \gamma_s}{\sum_{s'=1}^{S} \eta_{s',n,m} + \gamma_{s'}} \, t_{\nu_{s,n,m} - D + 1}\left(\mu_{s,n,m}, \frac{\Lambda_{s,n,m}(\kappa_{s,n,m} + 1)}{\kappa_{s,n,m}(\nu_{s,n,m} - D + 1)}\right). \tag{29}$$

Finally, given a sample of Ξ from this Markov chain, we can compute the conditional distribution over the stels Θ. These have a Dirichlet posterior distribution:

$$p(\Theta \mid \Xi, \gamma) = \prod_{n=1}^{N} p(\boldsymbol{\theta}_n \mid \{s_{n,m}\}_{m=1}^{M}, \gamma)$$
$$= \prod_{n=1}^{N} \frac{\Gamma\left(\sum_{s=1}^{S} \eta_{s,n} + \gamma_s\right)}{\prod_{s=1}^{S} \Gamma(\eta_{s,n} + \gamma_s)} \prod_{s=1}^{S} \theta_{s,n}^{\eta_{s,n}+\gamma_s-1}. \tag{30}$$

2.4 Estimating the Posterior Distribution over Stels

CIA is an unsupervised learning model of image statistics. For practical discriminative tasks, however, we wish to use CIA to provide informative features. We do this by constructing an estimate of the stel assignment probabilities Θ for each of the object classes we wish to identify.

The richest representation of the posterior distribution is achieved by constructing a mixture of Dirichlet distributions, where each component in the mixture is parameterized as in Eq. (30) and weighted equally:

$$p(\Theta \mid \gamma, \alpha) \approx \frac{1}{J} \sum_{j=1}^{J} \prod_{n=1}^{N} \frac{\Gamma\left(\sum_{s=1}^{S} \eta_{s,n}^{(j)} + \gamma_s\right)}{\prod_{s=1}^{S} \Gamma(\eta_{s,n}^{(j)} + \gamma_s)} \prod_{s=1}^{S} \theta_{s,n}^{\eta_{s,n}^{(j)}+\gamma_s-1}, \tag{31}$$

where $\eta_{s,n}^{(j)}$ denotes the jth sample of the assignments in the Markov chain from the previous section.

From the point of view of feature-extraction, however, it may be more practical to simply use a point estimate of Θ, denoted $\widehat{\Theta}$. One straightforward way to form such a point estimate is to average the predictive distributions arising from Eq. (30) as in

$$\hat{\theta}_{s,n} = \sum_{j=1}^{J} \frac{\eta_{s,n}^{(j)} + \gamma_s}{\sum_{s'=1}^{S} \eta_{s',n}^{(j)} + \gamma_{s'}}. \tag{32}$$

This uses the same $\eta_{s,n}^{(j)}$ samples as above. This point estimator is used for the experiments in this paper, although with the additional aspect that γ_s is set to zero for prediction. That is, the Dirichlet prior is used for training, but the predictions are not smoothed. This helps identify which stels are actually represented in the data.

An astute reader will notice, however, that stel indices are non-identifiable, but Eq. (32) implicitly assumes that stel indices are in fact identifiable across samples. However, we note that in practice, after a sufficient burn-in period, stel indices appear to not change in between samples due to the extremely slow mixing of the Gibbs sampler, and so practically, stel indices can be treated as identifiable across

samples. A theoretically valid approach would be that of analyzing statistics of the co-occurrence of stel membership assignments.

As we expect, some stels have a very small posterior for a given pixel. In fact, some stels have a very small posterior across all pixels in all images. These stels can be thought of the unused stels and their existence supports the notion that different classes need a different number of stels.

Given the per-pixel posterior distribution, we retain only the stels that are "in use" for the class. We define a stel to be in use if its total posterior distribution over pixels is greater than 0.02. That is, a stel s is in use if

$$\sum_{n=1}^{N} \hat{\theta}_{s,n} > 0.02. \tag{33}$$

If there are C classes of interest, indexed by c, then the posterior for each pixel n in class c is represented as an $S^{(c)}$-dimensional vector, where $S^{(c)}$ is the number of stels that are in use for class c.

Overall, the parameters we extract from the inference procedure across all classes are the collection of $\widehat{\Theta}^{(c)}$ distributions for every pixel in every class, where the dimension of $\widehat{\Theta}^{(c)}$ depends on the number of in use stels for class c. This set of parameters is the output of the CIA model that we will use in order to extract useful image representations for object recognition.

3 Using Stels for Supervised Learning Tasks

Stel models are density models of images and can be used for a wide variety of vision tasks. For image classification, stel models can be used to define class-conditional densities which are combined using Bayes' rule to classify test images, or stel models can be used to construct feature vectors that are fed into a discriminative learning algorithm. In Sect. 4, we report results on image classification using the latter approach. For this purpose, we construct feature vectors (image descriptors) in a way that is similar to the approach described in [24].

Stel models are used to define class-specific segmentations of images and those segmentations are used to construct feature vectors consisting of a histogram of SIFT codewords for each stel. In particular, we extract for any image (training or testing) a descriptor based on the $\widehat{\Theta}^{(c)}$ distributions described above. The descriptor can be easily used to calculate image similarities in a variety of ways. For example, a kernel operator that is based on the histogram intersection kernel [8] can be used to measure image similarity and a maximum-margin method such as the SVM can be used for classification.

We define a discrete set of K visual feature codewords $k \in \{1,2,\ldots,K\}$. For instance, to learn the codebook we can use the standard approach of extracting dense SIFT features and clustering them. We can then pre-process each image by computing per-pixel visual features $\{f_n \in \{1,2,\ldots,K\}\}_{n=1}^{N}$.

Now, given the feature index associated with each pixel n in the image, we construct a per-class, per-stel count histogram of visual features:

$$h_s^{(c)}(k) = \sum_{n=1}^{N} \hat{\theta}_{s,n}^{(c)} \delta(f_n, k). \tag{34}$$

We can define a concatenated histogram of features as

$$h^{(c)} = [h_1^{(c)}(1), \ldots, h_1^{(c)}(K), h_1^{(c)}(1), \ldots, h_2^{(c)}(K), \ldots, h_{S^{(c)}}^{(c)}(1), \ldots, h_{S^{(c)}}^{(c)}(K)], \tag{35}$$

where $S^{(c)}$ is the total number of stels used for object class c associated with the image. Thus, this representation gives a vector $h^{(c)} \in \mathbb{R}^{S^{(c)} \times K}$, and we obtain a set of C such vectors per image.

Using the histogram of features $h^{(c)}$, we define the class-specific similarity between two images, A and B, by the histogram intersection kernel [8]:

$$\mathrm{Ker}(A, B) = \min_k [h_A^{(c)}(k), h_B^{(c)}(k)]. \tag{36}$$

Note that using this stel kernel, one can better encode spatial relations. Rather than collecting histograms over arbitrarily defined quadrants as in [16], we collect histograms over stel segmentations, which provide useful spatial clues pertaining to the identity of the object.

An orthogonal representation to the CIA representation above is that of the spatial pyramid histogram of features [16], which is created by constructing a two-level spatial pyramid histogram of visual features over the four image quadrants, as well as the entire image. Note that the features can be created by a different codebook from the one used to create the $h^{(c)}$ collection.

We also define a combined descriptor, that is created by appending the aforementioned spatial pyramid descriptor to $h^{(c)}$. Thus, we obtain for every image a collection of C feature vectors, $h^{(c)} \in \mathbb{R}^{S^{(c)} \times K + 5K'}$, where the $5K'$ term comes from the spatial pyramid descriptor. In our experiments, we do use the same codebook, and $K = K'$. This representation can also be used in conjunction with the intersection kernel above to give a per-class kernel based similarity measure.

4 Experimental Evaluation

To evaluate the usefulness of our approach, we conducted several experiments, which we report in this section. We performed qualitative assessment by examining whether our new method, CIA, results in stel-based representations of images with varying levels of complexity as determined by the object class. We also performed quantitative analyses to determine whether CIA results in features that improve performance on object recognition tasks. Finally, we also are interested in how the behavior of CIA changes when the free parameters of the prior are adjusted. We investigated these properties using a subset of the Caltech101 image dataset.

4.1 Experimental Setup

4.1.1 Data

We selected a subset of 28 classes from the popular Caltech101 dataset, by identifying the 28 classes that had the largest number of examples per class. To balance class sizes, we chose 30 images per class randomly. The class labels are aeroplane, motorbike, background_Google, faces_easy, watch, leopard, bonsai, car_side, ketch, chandelier, hawksbill, grand_piano, brain, butterfly, helicopter, menorah, kangaroo, starfish, trilobite, buddha, ewer, sunflower, scorpion, revolver, laptop, ibis, llama and minaret. Multiple training-testing trials were used to obtain confidence intervals. In each trial, the 30 images in each class were randomly split into 15 training images and 15 test images. Experimental results are reported based on averaging ten such trials.

4.1.2 Image Preprocessing

The images were resized without cropping to be 50×50, and were converted to greyscale, with the intensities scaled to the interval $[0,1]$. This corresponds to $D = 1$ in Eq. (21). Note that images were *not* whitened (as is common in other vision approaches), since the CIA approach explicitly addresses invariance to colour.

4.1.3 Model Configuration

Unless otherwise specified, the maximum number of stels S was set to 12. The experiments used hyperparameters of $\gamma = 0.8$, $\Lambda_0 = 0.001$, $\mu_0 = 0.5$, $\kappa_0 = 0.05$, and $\nu_0 = 2.5$. The Gibbs sampler was used to generate 1200 samples, after burning in for 1800 iterations.

4.1.4 Visual Feature (SIFT) Codebook

Following the approach of [16], we extracted $100,000$ random SIFT features from the training set using code obtained from [26], and learnt a codebook of $K = 300$ visual codewords using K-means.

4.2 Learning a Flexible Number of Stels

Our first concern is whether the additional flexibility of the CIA model actually results in richer representations for different object classes. We gauged this by examining how many stels tend to be represented for each object class, when trained

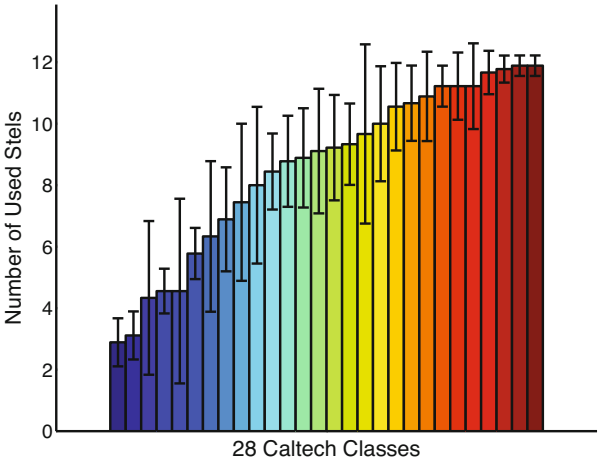


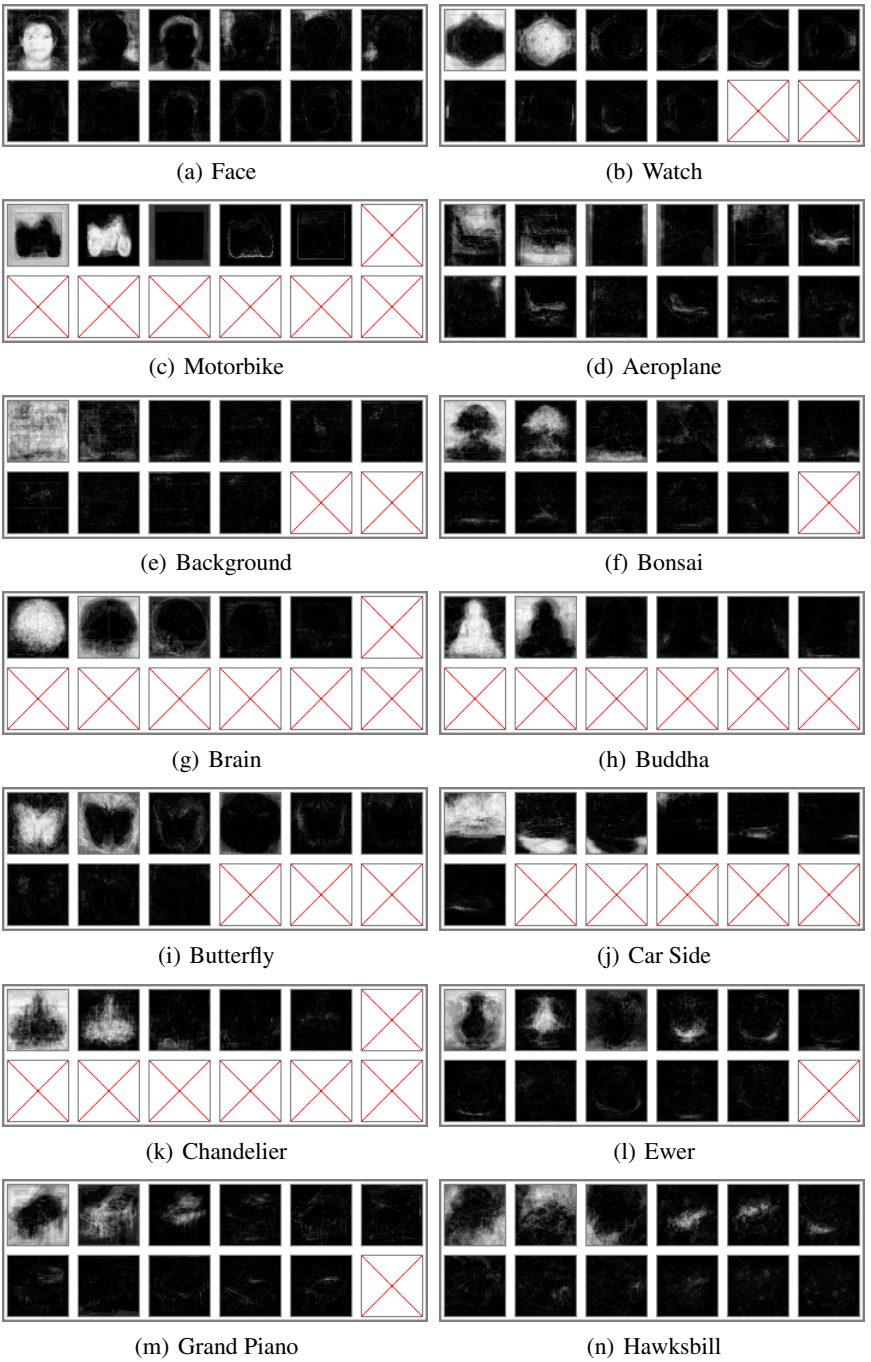
Fig. 7 A plot of the number of stels used by different Caltech classes. Error bars show the standard deviation and the classes are sorted by the mean number of stels used.

using the same hyperparameters. For each class, we trained a stel model separately and after training, the number of significant stels for each class, called 'stel usage', was determined using the threshold described in Eq. 33. For every class, this procedure was repeated ten times. We then examined the number of used stels per class across the different classes, and we report results averaged across trials.

Fig. 7 shows the mean and standard deviation of stel usage for each class, where the classes are sorted according to mean stel usage. Note that different classes use a different number of stels and this number varies widely across classes. The models that use the smallest number of stels on average correspond to the ibis, leopard and starfish classes, whereas models that use the largest number of stels on average correspond to the minaret, face and laptop classes.

Figs. 8 and 9 show stel probability maps extracted for every object class, using one of the random trials. We examine in more detail three representative cases: the face (Fig. 8(a)), watch (Fig. 8(b)), and motorbike (Fig. 8(c)) classes. As unused stels (as determined by the aforementioned thresholding) are not shown, it is clear that different objects are using different numbers of stels. One explanation for this may be that different classes have different intrinsic complexities in the parts and colours that comprise them. Alternatively, image classes that have many different poses, deformations, or background clutter may require more stels. Faces appear to be an example of this, since different stels are used to account for variations in expression, hair style and background.

For additional insight, we also examine the per-class histograms over the number of used stels, shown in Figs. 10 and 11. These histograms show the total number of above-threshold stels represented by all images in the class. The histograms were computed by aggregating all post burn-in samples from the Markov chain, across all images in the class and for all trials. The differences between the results illuminate



(a) Face (b) Watch

(c) Motorbike (d) Aeroplane

(e) Background (f) Bonsai

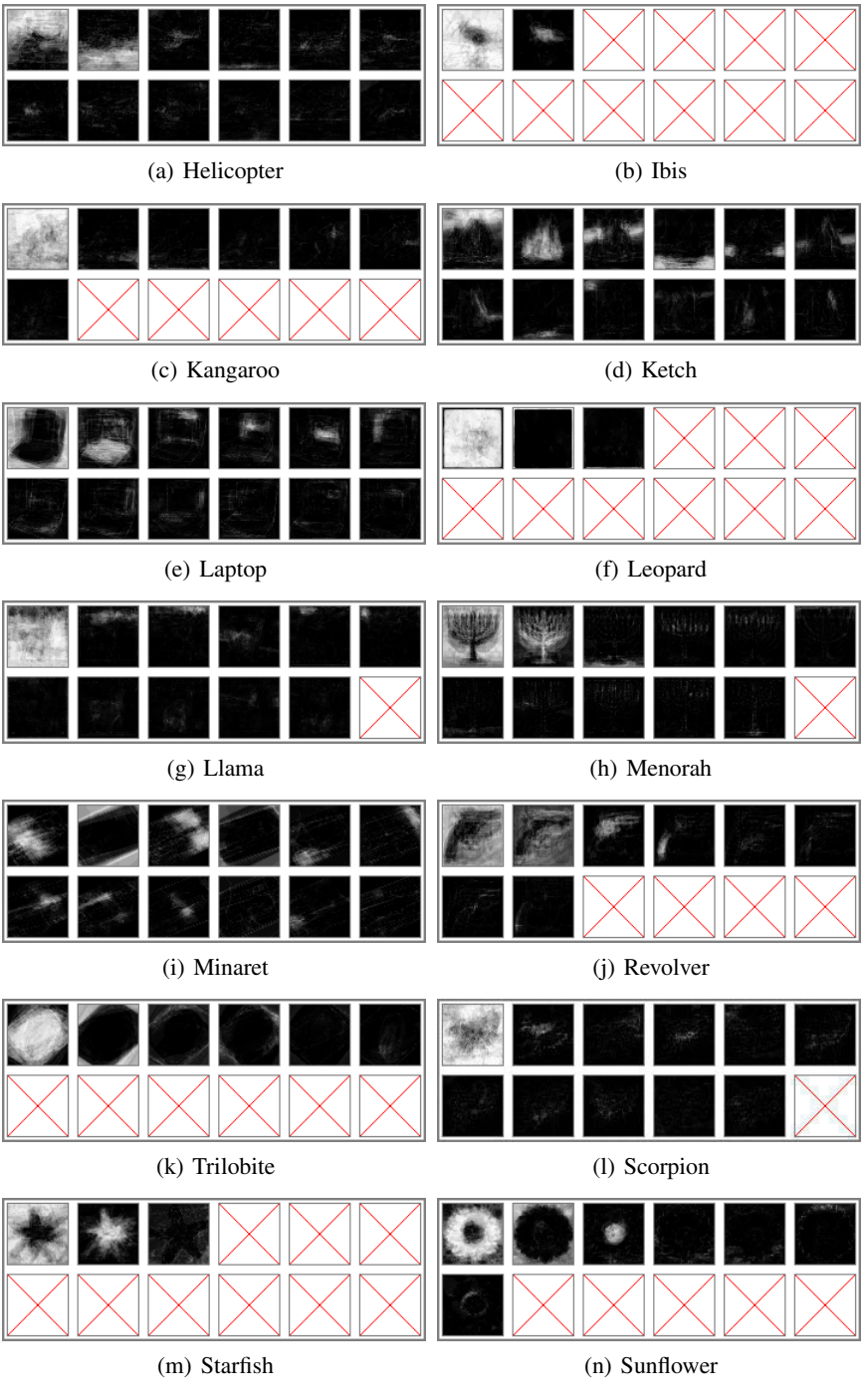
(g) Brain (h) Buddha

(i) Butterfly (j) Car Side

(k) Chandelier (l) Ewer

(m) Grand Piano (n) Hawksbill

Fig. 8 For each class, the stels learnt in a randomly selected trial are shown in order of decreasing probability mass. Some classes, such as faces, are modelled using a larger number of stels, while other classes, such as chandeliers, are modelled using a smaller number of stels.



(a) Helicopter (b) Ibis

(c) Kangaroo (d) Ketch

(e) Laptop (f) Leopard

(g) Llama (h) Menorah

(i) Minaret (j) Revolver

(k) Trilobite (l) Scorpion

(m) Starfish (n) Sunflower

Fig. 9 Continuation of Fig. 8

the flexibility of the CIA approach. In some classes, such as watch, the images are effectively modelled using only about four stels, while in other classes, such as face, the images typically require about eight stels.

4.3 Object Recognition Using CIA

In this section, we investigate whether our proposed method (CIA) can be used to extract useful representations for object recognition. For each of the 28 classes, we calculated the training-by-training, and training-by-test Gram matrices, based on the descriptors introduced in Sect. 3, and using the intersection kernel as defined in Eq. (36). Since there are 420 training images, and 420 test images in total, (28 classes with 15 images per class), each of these matrices is a 420×420 matrix. We trained a one-versus-all SVM classifier [25] using the 28 Gram matrices. These experiments used publicly-available code from [16, 4].

One motivation for our Bayesian method is to enable flexibility in the effective number of stels that are learnt. We hypothesized that this flexibility could lead to better representations and thus better performance on classification tasks. Our first comparison is therefore against the non-Bayesian counterpart for CIA, namely the basic non-Bayesian stel model [12]. For the basic stel model, it is necessary to set the number of stels in advance. This was done using leave-one-out cross-validation on the training set, allowing for between four and twelve stels. The number of stels was determined by the configuration which produced the best likelihood on held-out validation data.

Overall, the basic stel model with cross-validation obtained 42% classification accuracy, while our Bayesian method obtained 68% accuracy (see Table 1). Thus, we have strong evidence to support the conclusion that the additional flexibility leads to representations that are better for discrimination.

Next, we augmented the descriptor extracted using the Bayesian stel model with a standard two level spatial pyramid descriptor, to see if the combined descriptor would give improved performance. Including the spatial pyramid descriptor yields a modest improvement, increasing the classification accuracy to 72%. While the significance of this improvement is questionable (the standard deviation is about 2%), it may be explained by some classes not being very well modelled using the stel model. Some classes, such as the background_Google and leopard classes, are not well-segmented based on colour and shape co-occurrence. This prevents the stel features from providing useful information. In this case, collecting image features in the scheme of [16] provides the classifier with additional information that can be used to improve performance. One way to view collecting image features over a two-level spatial pyramid is that it augments our approach to fine-tune performance on segmentation-unfriendly classes.

Finally, we ask whether our descriptors are comparable to the descriptors typically used for object recognition, based on spatial pyramids. We used the SIFT descriptors over the image and image quadrants (the same representation we appended to our descriptor to obtain the augmented descriptor described above) in

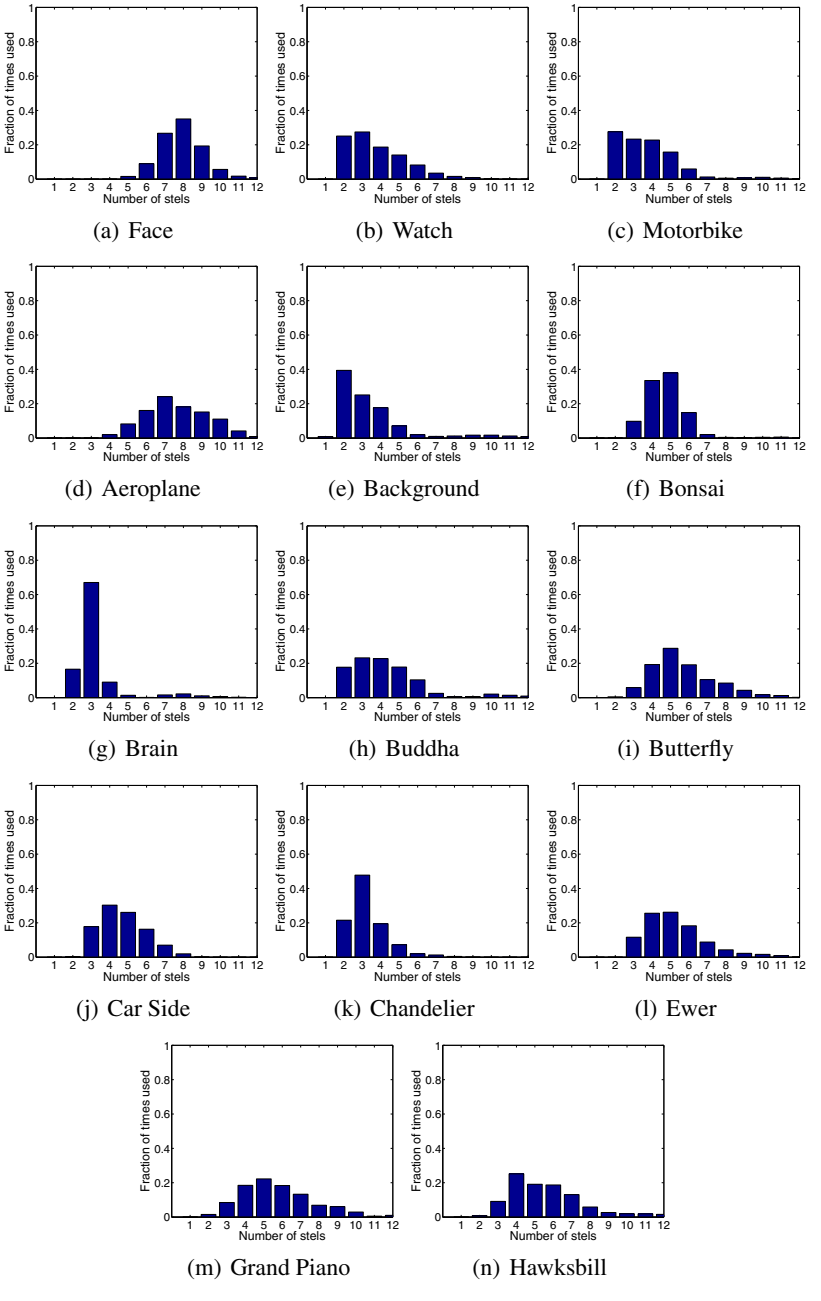


Fig. 10 Per-class histograms of the number of stels used for all images and all trials.

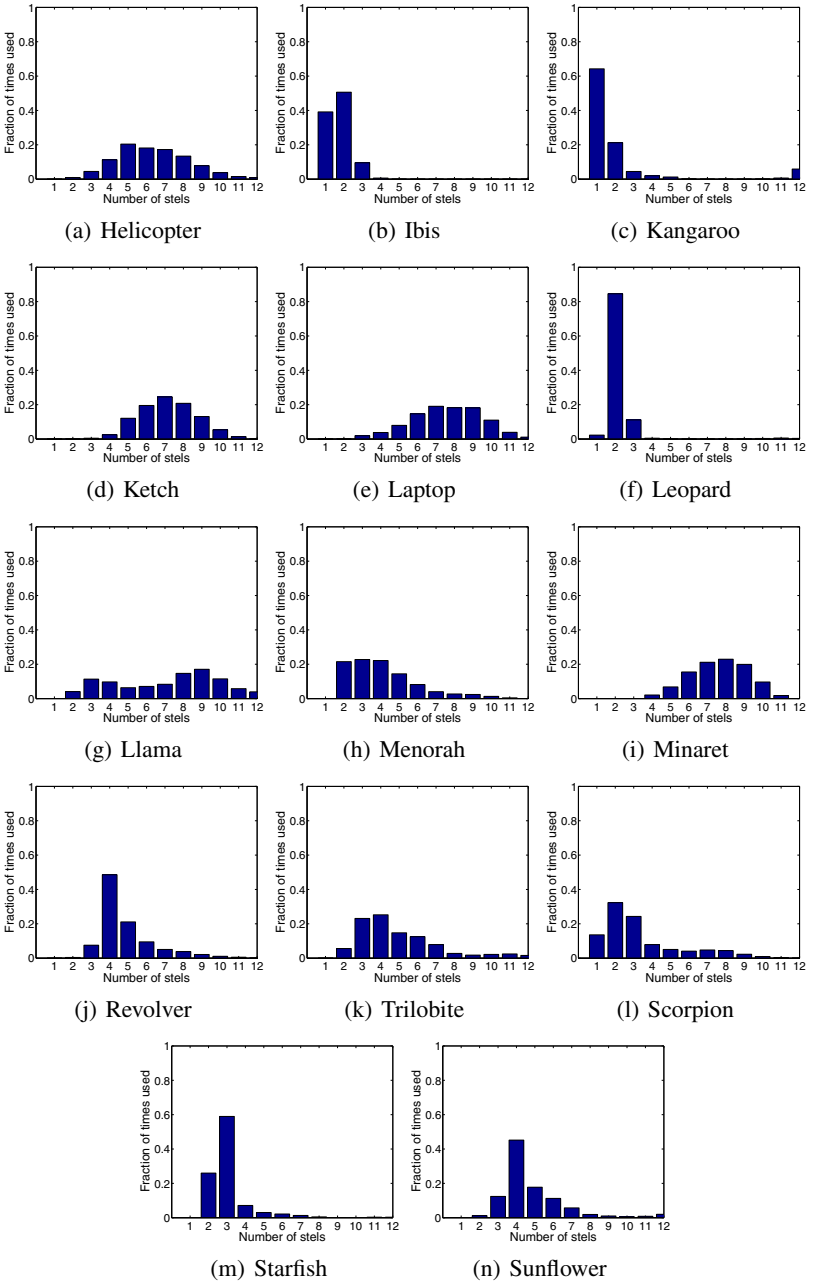


Fig. 11 Continuation of Fig. 10.

conjunction with the intersection kernel. The achieved accuracy was 71%, which is not statistically-significantly different from the result obtained using our method.

In summary, descriptors derived using our proposed method significantly improve on the basic stel model. In particular, our method discovers segmentations of object classes that are more conducive to object recognition. Additionally, our method is competitive with standard SIFT-based spatial pyramid methods, and provides a very different approach to defining feature vectors.

Table 1 Classification results on a subset of 28 classes from Caltech101. The first two entries were obtained using our proposed Bayesian method (CIA). The third entry follows the method of [15], using a fixed number of stels per class that was chosen using cross-validation. The fourth method is considered to be the state of the art and follows [16].

Method	Accuracy (sd $\sim 2\%$)
Bayesian stel model (CIA)	68%
Bayesian stel model with spatial pyramid	**72%**
Basic stel model with cross-validated numbers of stels	42%
SIFT with spatial pyramid	71%

4.4 Effect of Hyperparameter Choices on Learnt Stels

An important question in the context of Bayesian inference is the sensitivity of inferred models to the hyperparameter settings. While it is possible to infer or sample them, we can obtain intuition and a better understanding of the model by considering how the inferred model changes as a function of the hyperparameter settings.

In this section we report some of our findings when trying different parameter settings for the greyscale version of the model. As the images are greyscale with values in $[0,1]$, the hyperparameter value for the mean, μ_0, was not expected to be particularly sensitive, and we set it throughout all experiments to be $\mu_0 = 0.5$. However, it was expected that the other normal-inverse Wishart parameters would have an effect on the results. In the following experiments, we centred the hyperparameters at a "reasonable" configuration and then perturbed them one at a time and to examine the effects. The centre values are $\Lambda_0 = 0.001$, $\mu_0 = 0.5$, $\kappa_0 = 0.1$, $\nu_0 = 3$. We examine the results of perturbing κ_0, ν_0, and Λ_0.

4.4.1 Variation in κ_0

In this set of experiments we considered the values $\kappa_0 \in \{0.01, 0.1, 1\}$. Results are shown in Fig. 12. We observe that for $\kappa_0 = 1$, we obtain many similar stels that tend to be below the threshold. As we lower κ_0, we allow larger variations from the prior mean. For represented stels with many pixels, however, the prior mean

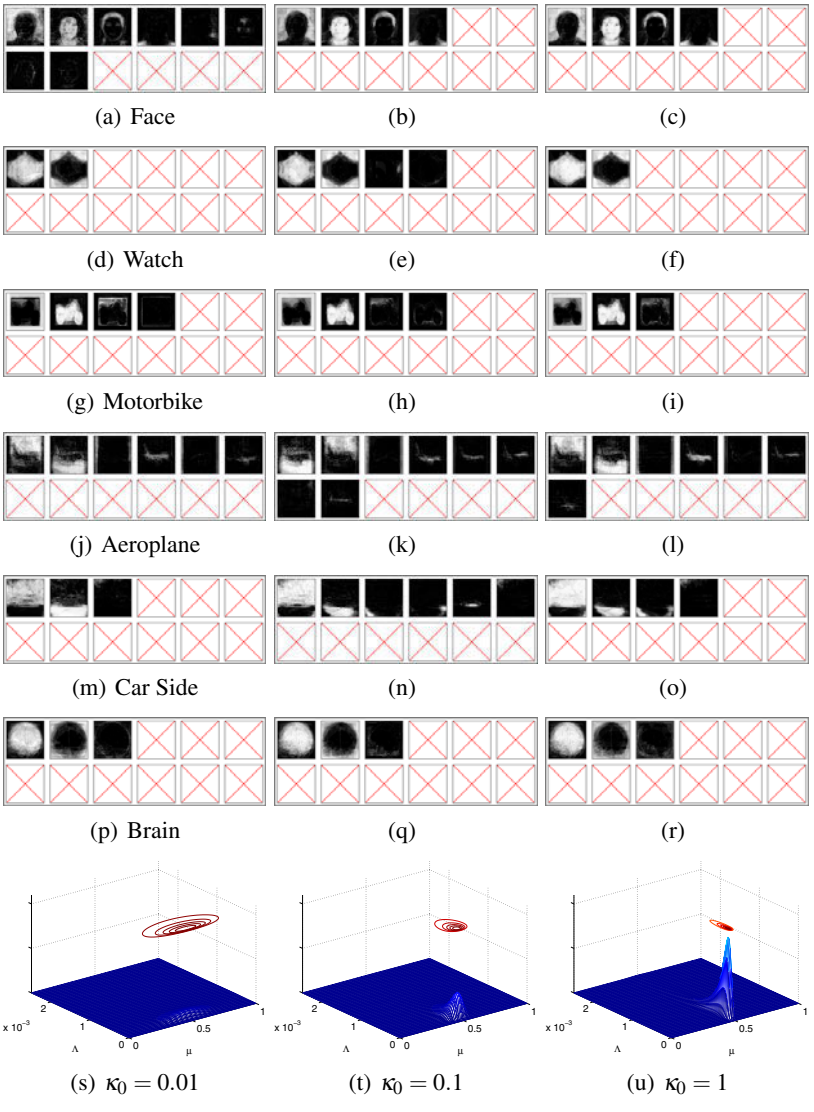


Fig. 12 Inferred stels for different settings of κ_0 across columns. The other NIW parameters are set to $\{\mu_0 = 0.5, \Lambda_0 = 0.001, \nu_0 = 3\}$

has little effect; κ_0 primarily changes the default colour distributions for below-threshold stels. Figs. 12(s)-12(u) show contour plots for the normal-inverse Wishart prior distribution on the mean and variance parameter of the colour model, for each setting of κ_0.

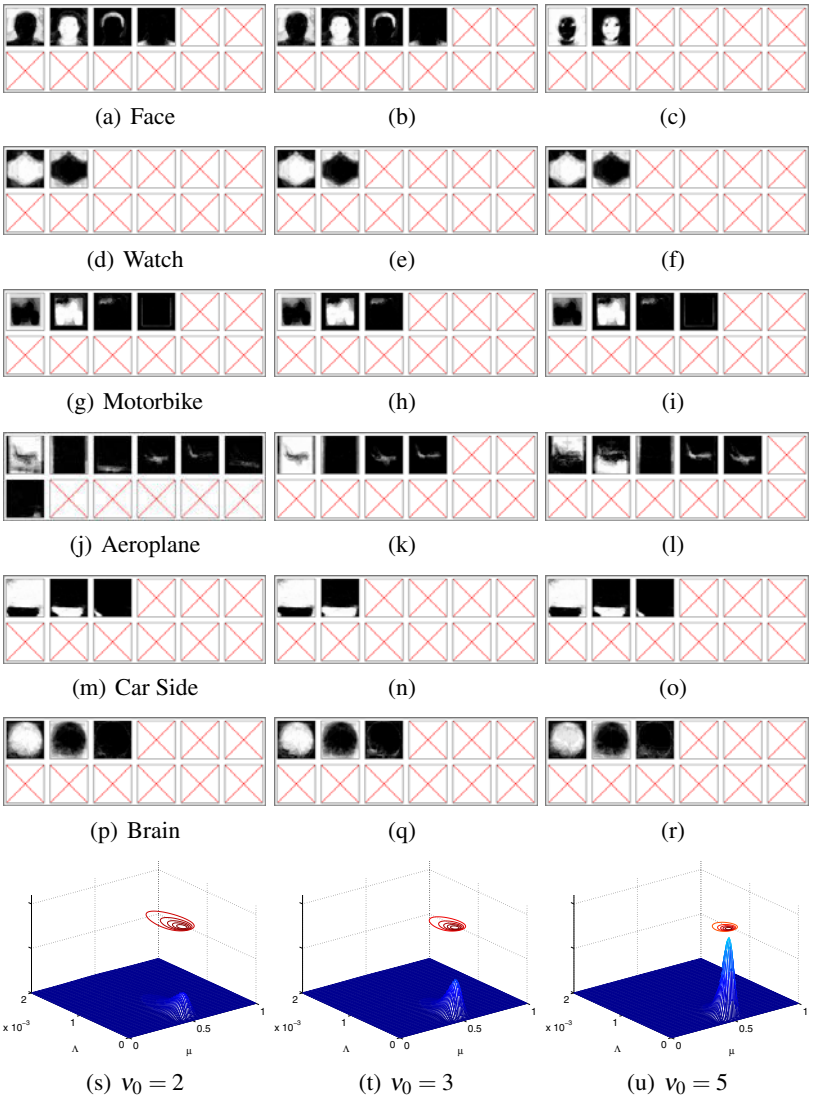


Fig. 13 Inferred stels for different settings of ν_0 across columns. The other NIW parameters are set to $\{\mu_0 = 0.5, \Lambda_0 = 0.001, \kappa_0 = 0.1\}$

4.4.2 Variation in ν_0

This set of experiments considered the values $\nu_0 \in \{2,3,5\}$. Results are shown in Fig. 13. Increasing ν_0 increases the effect of the prior covariance Λ_0. Given the prior choice of Λ_0, this results in narrower colour distributions and decreased flexibility in what pixels each stel can explain. For example, in the faces shown in Fig. 13(c), a

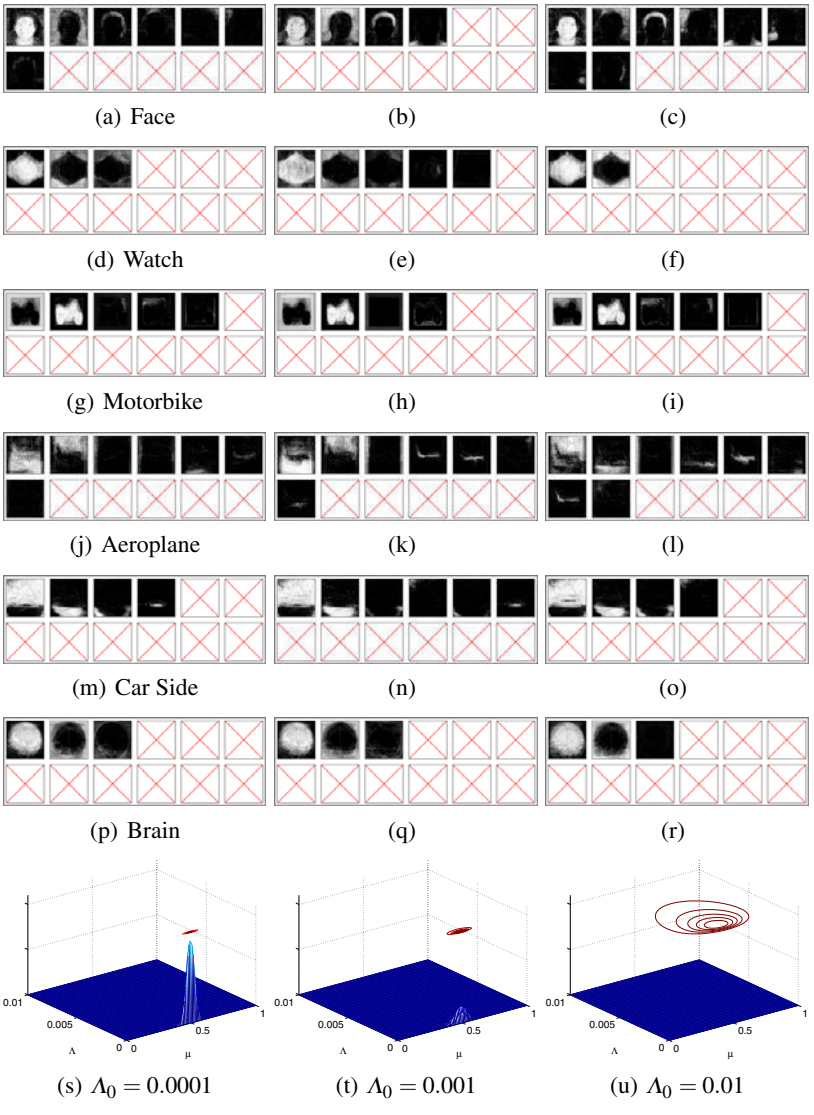


Fig. 14 Inferred stels for different settings of Λ_0 across columns. The other NIW parameters are set to $\{\mu_0 = 0.5, \kappa_0 =, \nu_0 = 3\}$

single stel cannot explain both the eyes and the face. Figs. 13(s)-13(u), show contour plots for the normal-inverse Wishart prior distribution on the mean and variance parameters of the colour model, as ν_0 is varied.

4.4.3 Variation in Λ_0

These experiments considered the values $\Lambda \in \{0.0001, 0.001, 0.01, \}$. Results are shown in Fig. 14. This hyperparameter corresponds to the prior on the covariance. Although this has a large effect on the normal-inverse-Wishart prior, there is little sensitivity evident in the learnt stels. Figs. 14(s)-14(s) show contour plots for the normal-inverse Wishart prior distribution on the mean and variance parameters of the colour model as Λ_0 is varied.

5 Summary

We have introduced a novel generative Bayesian framework, the colour-invariant admixture model (CIA), for generalizing stel models. This rectifies a previous weakness in stel-based models, in which it is difficult to regularize the distribution over stels. Our admixture-based approach represents the full posterior distribution over stel parameters, enabling different numbers of stels to be represented to capture variation in object complexity. We have also introduced a straightforward Gibbs sampler for performing inference in this model. Our empirical analyses demonstrate that CIA is capable of learning varying complexity in stels for each class. Additionally, CIA outperforms stel modelling approaches that require cross-validation, and performs comparably to the popular SIFT-based pyramid matching.

CIA can be used to segment images in an unsupervised fashion, and we have shown that this segmentation provides a rich backdrop on which to perform object recognition. It robustly captures spatial relations between features, a crucial piece of any foundation for state-of-the-art object recognition.

Although our approach provides a way of inferring a class-specific distribution over the stels, our approach may still be sensitive the maximum number of stels S. For instance, allowing a large number of stels to be used may result in "noisy" stels, in which many stels are speckly and do not appear to represent meaningful structure. This free parameter is a difficult-to-avoid side-effect of this generative model. One future direction for resolving this difficulty is to use a Bayesian nonparametric approach in which an unbounded number of stels are allowed by the model, using, for example, the hierarchical Dirichlet process [30].

In addition, patch-based models, such as convolutional neural networks, have recently become popular in the vision community. An extension to this work is to model an image as a collection of smaller patches, all of which use the same palette to colour the image. Here, a vocabulary of patches, similar to Gabor filters, could be learnt, and the types of patches used in an image could be used to discriminate one class from another. For example, one patch from the vocabulary may represent the wheel of a motorbike, and so this type of patch being present twice in an image could help distinguish between motorbikes and sunflowers, which have no wheels.

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
2. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Recognition and Machine Intelligence 8(6), 679–698 (1986)
3. Cao, L., Li, F.-F.: Spatially coherent latent topic model for concurrent object segmentation and classification. In: Proceedings of the Eleventh IEEE International Conference on Computer Vision (2007)
4. Chang, C., Lin, C.: LIBSVM: a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
5. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? Journal of Machine Learning Research 11, 625–660 (2010)
6. Frey, B.J., Jojic, N.: Estimating mixture models of images and inferring spatial transformations using the EM algorithm. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 416–422 (1999)
7. Frey, B.J., Jojic, N.: Transformation-invariant clustering using the EM algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(1) (2003)
8. Grauman, K., Darrell, T.: The pyramid match kernel: Discriminative classification with sets of image features. In: Proceedings of the Tenth IEEE International Conference on Computer Vision, pp. 1458–1465 (2005)
9. Grenander, U.: Lectures in Pattern Theory I, II and III: Pattern Analysis, Pattern Synthesis and Regular Structures. Springer, Berlin (1976-1981)
10. Hinton, G.E.: Connectionist learning procedures. Artificial Intelligence 40, 185–234 (1989)
11. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)
12. Jojic, N., Caspi, Y.: Capturing image structure with probabilistic index maps. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 212–219 (2004)
13. Jojic, N., Frey, B.J.: Learning flexible sprites in video layers. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2001)
14. Jojic, N., Perina, A., Cristani, M., Murino, V., Frey, B.J.: Stel component analysis: Modeling spatial correlations in image class structure. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 2044–2051 (2009)
15. Jojic, N., Perina, A., Cristani, M., Murino, V., Frey, B.J.: Stel component analysis: Modeling spatial correlations in image class structure. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2044–2051 (2009)
16. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2169–2178 (2006)
17. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
18. Li, F.-F., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. In: Proceedings of the IEEE CVPR Workshop on Generative Model Based Vision, p. 178 (2004)
19. Lowe, D.: Object recognition from local scale-invariant features. In: Proceedings of the Seventh IEEE International Conference on Computer Vision (1999)
20. Marr, D.: Vision: A computational investigation into human representation and processing of visual information. W. H. Freeman and Company, San Franciso (1982)
21. Mumford, D.: Neuronal architectures for pattern-theoretic problems. In: Koch, C., Davis, J. (eds.) Large-Scale Theories of the Cortex, pp. 125–152. MIT Press, Cambridge (1994)

22. Murphy, K.P.: Conjugate Bayesian analysis of the Gaussian distribution. Technical report. University of British Columbia (2007)
23. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381, 607–609 (1996)
24. Perina, A., Jojic, N., Castellani, U., Cristani, M., Murino, V.: Object Recognition with Hierarchical Stel Models. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6316, pp. 15–28. Springer, Heidelberg (2010)
25. Rifkin, R., Klautau, A.: In defense of one-vs-all classification. Journal of Machine Learning Research 5, 101–141 (2004)
26. Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: Labelme: a database and web-based tool for image annotation. International Journal of Computer Vision 77, 157–173 (2008)
27. Serre, T., Oliva, A., Poggio, T.: A feedforward architecture accounts for rapid categorization. Proceedings of the National Academy of Sciences 104(15), 6424–6429 (2007)
28. Sivic, J., Russell, B.C., Efros, A.A., Zisserman, A., Freeman, W.T.: Discovering objects and their locations in images. In: Proceedings of the Tenth IEEE International Conference on Computer Vision (2005)
29. Sudderth, E.B., Torralba, A., Freeman, W.T., Willsky, A.S.: Describing visual scenes using transformed object parts. International Journal of Computer Vision 77 (2008)
30. Teh, Y.W., Jordan, M.I., Beal, M.J., Blei, D.M.: Hierarchical Dirichlet processes. Journal of the American Statistical Association 101(476), 1566–1581 (2006)
31. Zhu, S.C., Mumford, D.: GRADE: Gibbs reaction and diffusion equations — a framework for pattern synthesis, image denoising, and removing clutter. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (1998)

Real-Time Human Pose Recognition in Parts from Single Depth Images

Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp,
Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake

Abstract. This chapter describes a method to quickly and accurately predict 3D positions of body joints from a single depth image, using no temporal information. We take an object recognition approach, designing an intermediate body parts representation that maps the difficult pose estimation problem into a simpler per-pixel classification problem. Our large and highly varied training dataset allows the classifier to estimate body parts invariant to pose, body shape, clothing, *etc*.. Finally we generate confidence-scored 3D proposals of several body joints by reprojecting the classification result into world space and finding local modes of a 3D non-parametric density. The system runs at around 200 frames per second on consumer hardware. Our evaluation shows high accuracy on both synthetic and real test sets, and investigates the effect of several training parameters. We achieve state of the art accuracy in our comparison with related work and demonstrate improved generalization over exact whole-skeleton nearest neighbor matching.

1 Introduction

Robust interactive human body tracking has applications including gaming, human-computer interaction, security, telepresence, and even health-care. The task has recently been greatly simplified by the introduction of real-time depth cameras [16, 19, 44, 37, 28, 13]. However, even the best existing systems still exhibit limitations. In particular, until the launch of Kinect [21], none ran at interactive rates on consumer hardware while handling a full range of human body shapes and sizes undergoing general body motions. Some systems achieve high speeds by tracking from frame to frame but struggle to re-initialize quickly and so are not robust. In this chapter, we focus on pose recognition in parts: detecting from a single depth image a small set of 3D position candidates for each skeletal joint. Our focus on

Jamie Shotton · Andrew Fitzgibbon · Mat Cook · Toby Sharp ·
Mark Finocchio · Richard Moore · Alex Kipman · Andrew Blake
Microsoft Research Cambridge and Xbox Incubation

R. Cipolla et al. (Eds.): Machine Learning for Computer Vision, SCI 411, pp. 119–135.
springerlink.com

per-frame initialization and recovery is designed to complement any appropriate tracking algorithm [7, 39, 16, 42, 13] that might further incorporate temporal and kinematic coherence. The algorithm presented here forms a core component of the Kinect gaming platform [21].

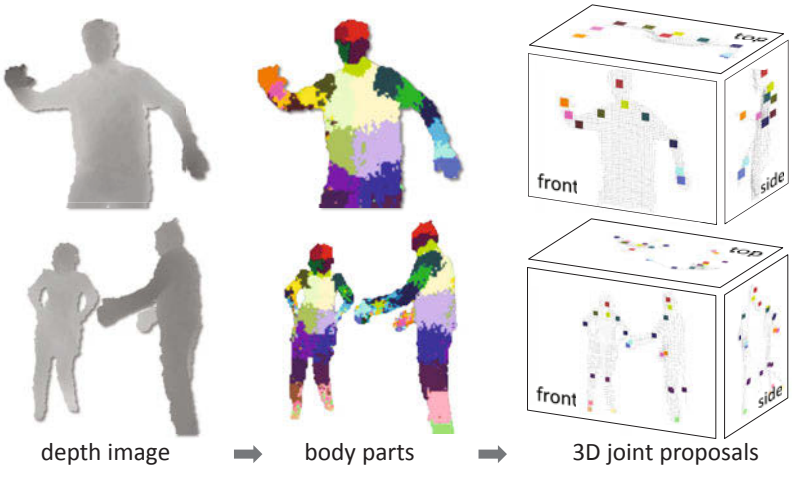


Fig. 1 Overview. From an single input depth image, a per-pixel body part distribution is inferred. (Colors indicate the most likely part labels at each pixel, and correspond in the joint proposals). Local modes of this signal are estimated to give high-quality proposals for the 3D locations of body joints, even for multiple users.

Illustrated in Fig. 1 and inspired by recent object recognition work that divides objects into parts (*e.g.,* [12, 43]), our approach is driven by two key design goals: computational efficiency and robustness. A single input depth image is segmented into a dense probabilistic body part labeling, with the parts defined to be spatially localized near skeletal joints of interest. Reprojecting the inferred parts into world space, we localize spatial modes of each part distribution and thus generate (possibly several) confidence-weighted proposals for the 3D locations of each skeletal joint.

We treat the segmentation into body parts as a per-pixel classification task (no pairwise terms or CRF have proved necessary). Evaluating each pixel separately avoids a combinatorial search over the different body joints, although within a single part there are of course still dramatic differences in the contextual appearance. For training data, we generate realistic synthetic depth images of humans of many shapes and sizes in highly varied poses sampled from a large motion capture database. We train a deep randomized decision forest classifier which avoids overfitting by using hundreds of thousands of training images. Simple, discriminative depth comparison image features yield 3D translation invariance while maintaining high computational efficiency. For further speed, the classifier can be run in parallel on each pixel on a GPU [34]. Finally, spatial modes of the inferred per-pixel distributions are computed using mean shift [10] resulting in the 3D joint proposals.

An optimized implementation of our algorithm runs in under 5ms per frame (200 frames per second) on the Xbox 360 GPU, at least one order of magnitude faster than existing approaches. It works frame-by-frame across dramatically differing body shapes and sizes, and the learned discriminative approach naturally handles self-occlusions and poses cropped by the image frame. We evaluate on both real and synthetic depth images, containing challenging poses of a varied set of subjects. Even without exploiting temporal or kinematic constraints, the 3D joint proposals are both accurate and stable. We investigate the effect of several training parameters and show how very deep trees can still avoid overfitting due to the large training set. We demonstrate that our part proposals generalize at least as well as exact nearest-neighbor in both an idealized and realistic setting, and show a substantial improvement over the state of the art. Further, results on silhouette images suggest more general applicability of our approach.

Our main contribution is to treat pose estimation as object recognition using a novel intermediate body parts representation designed to spatially localize joints of interest at low computational cost and high accuracy. Our experiments also carry several insights: (i) synthetic depth training data is an excellent proxy for real data; (ii) scaling up the learning problem with varied synthetic data is important for high accuracy; and (iii) our parts-based approach generalizes better than even an oracular exact nearest neighbor.

1.1 Related Work

Human pose estimation has generated a vast literature (surveyed in [22, 29]). The recent availability of depth cameras has spurred further progress [16, 19, 28]. Grest *et al.* [16] use Iterated Closest Point to track a skeleton of a known size and starting position. Anguelov *et al.* [3] segment puppets in 3D range scan data into head, limbs, torso, and background using spin images and a MRF. In [44], Zhu & Fujimura build heuristic detectors for coarse upper body parts (head, torso, arms) using a linear programming relaxation, but require a T-pose initialization to size the model. Siddiqui & Medioni [37] hand craft head, hand, and forearm detectors, and show data-driven MCMC model fitting outperforms ICP. Kalogerakis *et al.* [18] classify and segment vertices in a full closed 3D mesh into different parts, but do not deal with occlusions and are sensitive to mesh topology. Most similar to our approach, Plagemann *et al.* [28] build a 3D mesh to find geodesic extrema interest points which are classified into 3 parts: head, hand, and foot. Their method provides both a location and orientation estimate of these parts, but does not distinguish left from right and the use of interest points limits the choice of parts.

Advances have also been made using conventional intensity cameras, though typically at much higher computational cost. Bregler & Malik [7] track humans using twists and exponential maps from a known initial pose. Ioffe & Forsyth [17] group parallel edges as candidate body segments and prune combinations of segments using a projected classifier. Mori & Malik [24] use the shape context descriptor to match exemplars. Ramanan & Forsyth [31] find candidate body segments as pairs

of parallel lines, clustering appearances across frames. Shakhnarovich *et al.* [33] estimate upper body pose, interpolating k-NN poses matched by parameter sensitive hashing. Agarwal & Triggs [1] learn a regression from kernelized image silhouettes features to pose. Sigal *et al.* [39] use eigen-appearance template detectors for head, upper arms and lower legs proposals. Felzenszwalb & Huttenlocher [11] apply pictorial structures to estimate pose efficiently. Navaratnam *et al.* [25] use the marginal statistics of unlabeled data to improve pose estimation. Urtasun & Darrel [41] proposed a local mixture of Gaussian Processes to regress human pose. Auto-context was used in [40] to obtain a coarse body part labeling but this was not defined to localize joints and classifying each frame took about 40 seconds. Rogez *et al.* [32] train randomized decision forests on a hierarchy of classes defined on a torus of cyclic human motion patterns and camera angles. Wang & Popović [42] track a hand clothed in a colored glove. Our system could be seen as automatically inferring the colors of an virtual colored suit from a depth image. Bourdev & Malik [6] present 'poselets' that form tight clusters in both 3D pose and 2D image appearance, detectable using SVMs.

Fig. 2 Synthetic and real data. Pairs of depth image and ground truth body parts. Note wide variety in pose, shape, clothing, and crop.

2 Data

Pose estimation research has often focused on techniques to overcome lack of training data [25], because of two problems. First, generating realistic intensity images using computer graphics techniques [33, 27, 26] is hampered by the huge color and texture variability induced by clothing, hair, and skin, often meaning that the data are reduced to 2D silhouettes [1]. Although depth cameras significantly reduce this difficulty, considerable variation in body and clothing *shape* remains. The second limitation is that synthetic body pose images are of necessity fed by motion-capture (mocap) data. Although techniques exist to simulate human motion (*e.g.,* [38]) they do not yet produce the range of volitional motions of a human subject.

In this section we review depth imaging and show how we use real mocap data, retargetted to a variety of base character models, to synthesize a large, varied dataset. We believe this dataset to considerably advance the state of the art in both scale and variety, and demonstrate the importance of such a large dataset in our evaluation.

2.1 Depth Imaging

Depth imaging technology has advanced dramatically over the last few years, finally reaching a consumer price point with the launch of Kinect [21]. Pixels in a depth image indicate calibrated depth in the scene, rather than a measure of intensity or color. We employ the Kinect camera which gives a 640x480 image at 30 frames per second with depth resolution of a few centimeters.

Depth cameras offer several advantages over traditional intensity sensors, working in low light levels, giving a calibrated scale estimate, being color and texture invariant, and resolving silhouette ambiguities in pose. They also greatly simplify the task of background subtraction which we assume in this work. But most importantly for our approach, it is straightforward to synthesize realistic depth images of people and thus build a large training dataset cheaply.

2.2 Motion Capture Data

The human body is capable of an enormous range of poses which are difficult to simulate. Instead, we capture a large database of motion capture (mocap) of human actions. Our aim was to span the wide variety of poses people would make in an entertainment scenario. The database consists of approximately 500k frames in a few hundred sequences of driving, dancing, kicking, running, navigating menus, *etc*..

We expect our semi-local body part classifier to *generalize* somewhat to unseen poses. In particular, we need not record all possible combinations of the different limbs; in practice, a wide range of poses proves sufficient. Further, we need not record mocap with variation in rotation about the vertical axis, mirroring left-right, scene position, body shape and size, or camera pose, all of which can be added in (semi-)automatically.

Since the classifier uses no temporal information, we are interested only in static *poses* and not motion. Often, changes in pose from one mocap frame to the next are so small as to be insignificant. We thus discard many similar, redundant poses from the initial mocap data using 'furthest neighbor' clustering [15] where the distance between poses $\mathbf{X}_1$ and $\mathbf{X}_2$ is defined as $\max_j \|\mathbf{X}_1^j - \mathbf{X}_2^j\|_2$, the maximum Euclidean distance over body joints j. We use a subset of 100k poses such that no two poses are closer than 5cm.

We have found it necessary to iterate the process of motion capture, sampling from our model, training the classifier, and testing joint prediction accuracy in order to refine the mocap database with regions of pose space that had been previously missed out. Our early experiments employed the CMU mocap database [9] which gave acceptable results though covered far less of pose space.

2.3 Generating Synthetic Data

We build a randomized rendering pipeline from which we can sample fully labeled training images. Our goals in building this pipeline were twofold: realism and variety. For the learned model to work well, the samples must closely resemble real camera images, and contain good coverage of the appearance variations we hope to recognize at test time. While depth/scale and translation variations are handled explicitly in our features (see below), other invariances cannot be encoded efficiently. Instead we learn invariance from the data to camera pose, body pose, and body size and shape.

The synthesis pipeline first randomly samples a set of parameters, and then uses standard computer graphics techniques to render depth and (see below) body part images from texture mapped 3D meshes. The mocap is retargetting to each of 15 base meshes spanning the range of body shapes and sizes, using [4]. Further slight random variation in height and weight give extra coverage of body shapes. Other randomized parameters include the mocap frame, camera pose, camera noise, clothing and hairstyle. We provide more details of these variations in the supplementary material. Fig. 2 compares the varied output of the pipeline to hand-labeled real camera images.

3 Body Part Inference and Joint Proposals

In this section we describe our intermediate body parts representation, detail the discriminative depth image features, review decision forests and their application to body part recognition, and finally discuss how a mode finding algorithm is used to generate joint position proposals.

3.1 Body Part Labeling

A key contribution of this work is our intermediate body part representation. We define several localized body part labels that densely cover the body, as color-coded in Fig. 2. Some of these parts are defined to directly localize particular skeletal joints of interest, while others fill the gaps or could be used in combination to predict other joints. Our intermediate representation transforms the problem into one that can readily be solved by efficient classification algorithms; we show in 4.3 that the penalty paid for this transformation is small.

The parts are specified in a texture map that is retargetted to skin the various characters during rendering. The pairs of depth and body part images are used as fully labeled data for learning the classifier (see below). For the experiments in this chapter, we use 31 body parts: LU/RU/LW/RW head, neck, L/R shoulder, LU/RU/LW/RW arm, L/R elbow, L/R wrist, L/R hand, LU/RU/LW/RW torso, LU/RU/LW/RW leg, L/R

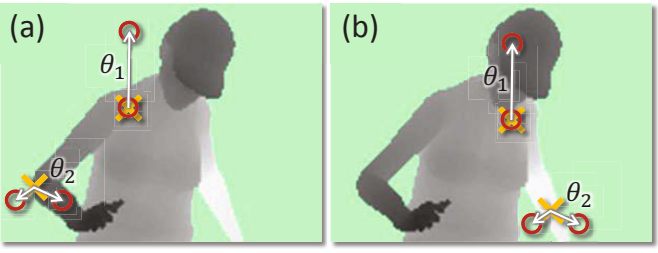


Fig. 3 Depth image features. The yellow crosses indicates the pixel $\mathbf{x}$ being classified. The red circles indicate the offset pixels as defined in Eq. 1. In (a), the two example features give a large depth difference response. In (b), the same two features at new image locations give a much smaller response.

knee, L/R ankle, L/R foot (Left, Right, Upper, loWer). Distinct parts for left and right allow the classifier to disambiguate the left and right sides of the body.

Of course, the precise definition of these parts could be changed to suit a particular application. For example, in an upper body tracking scenario, all the lower body parts could be merged. Parts should be sufficiently small to accurately localize body joints, but not too numerous as to waste capacity of the classifier.

3.2 Depth Image Features

We employ simple depth comparison features, inspired by those in [20]. At a given pixel $\mathbf{x}$, the features compute

$$f_\theta(I,\mathbf{x}) = d_I\left(\mathbf{x}+\frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x}+\frac{\mathbf{v}}{d_I(\mathbf{x})}\right), \tag{1}$$

where $d_I(\mathbf{x})$ is the depth at pixel $\mathbf{x}$ in image I, and parameters $\theta = (\mathbf{u},\mathbf{v})$ describe offsets $\mathbf{u}$ and $\mathbf{v}$. The normalization of the offsets by $\frac{1}{d_I(\mathbf{x})}$ ensures the features are depth invariant: at a given point on the body, a fixed *world space* offset will result whether the pixel is close or far from the camera. The features are thus 3D translation invariant (modulo perspective effects). If an offset pixel lies on the background or outside the bounds of the image, the depth probe $d_I(\mathbf{x}')$ is given a large positive constant value.

Fig. 3 illustrates two features at different pixel locations $\mathbf{x}$. Feature f_{θ_1} looks upwards: Eq. 1 will give a large positive response for pixels $\mathbf{x}$ near the top of the body, but a value close to zero for pixels $\mathbf{x}$ lower down the body. Feature f_{θ_2} may instead help find thin vertical structures such as the arm.

Individually these features provide only a weak signal about which part of the body the pixel belongs to, but in combination in a decision forest they are sufficient to accurately disambiguate all trained parts. The design of these features was

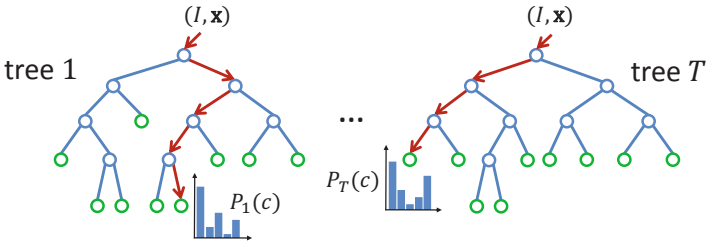


Fig. 4 Randomized Decision Forests. A forest is an ensemble of trees. Each tree consists of split nodes (blue) and leaf nodes (green). The red arrows indicate the different paths that might be taken by different trees for a particular input.

strongly motivated by their computational efficiency: no preprocessing is needed; each feature need only read at most 3 image pixels and perform at most 5 arithmetic operations; and the features can be straightforwardly implemented on the GPU. Given a larger computational budget, one could employ potentially more powerful features based on, for example, depth integrals over regions, curvature, or local descriptors *e.g.,* [5].

3.3 Randomized Decision Forests

Randomized decision trees and forests [35, 30, 2, 8] have proven fast and effective multi-class classifiers for many tasks [20, 23, 36], and can be implemented efficiently on the GPU [34]. As illustrated in Fig. 4, a forest is an ensemble of T decision trees, each consisting of split and leaf nodes. Each split node consists of a feature f_θ and a threshold τ. To classify pixel $\mathbf{x}$ in image I, one starts at the root and repeatedly evaluates Eq. 1, branching left or right according to the comparison to threshold τ. At the leaf node reached in tree t, a learned distribution $P_t(c|I,\mathbf{x})$ over body part labels c is stored. The distributions are averaged together for all trees in the forest to give the final classification

$$P(c|I,\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|I,\mathbf{x}) \ . \tag{2}$$

Training. Each tree is trained on a different set of randomly synthesized images. A random subset of 2000 example pixels from each image is chosen to ensure a roughly even distribution across body parts. Each tree is trained using the following algorithm [20]:

1. Randomly propose a set of splitting candidates $\phi = (\theta, \tau)$ (feature parameters θ and thresholds τ).
2. Partition the set of examples $Q = \{(I,\mathbf{x})\}$ into left and right subsets by each ϕ:

$$Q_\mathrm{l}(\phi) = \{\ (I,\mathbf{x}) \mid f_\theta(I,\mathbf{x}) < \tau\ \} \tag{3}$$

$$Q_\mathrm{r}(\phi) = Q \setminus Q_\mathrm{l}(\phi) \tag{4}$$

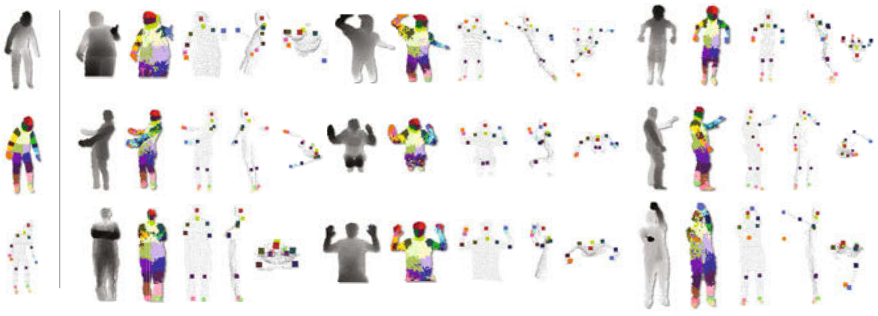


Fig. 5 Example inferences. Synthetic (top row); real (middle); failure modes (bottom). Left column: ground truth for a neutral pose as a reference. In each example we see the depth image, the inferred most likely body part labels, and the joint proposals show as front, right, and top views (overlaid on a depth point cloud). Only the most confident proposal for each joint above a fixed, shared threshold is shown.

3. Compute the ϕ giving the largest gain in information:

$$\phi^\star = \underset{\phi}{\operatorname{argmax}}\, G(\phi) \tag{5}$$

$$G(\phi) = H(Q) - \sum_{s\in\{\mathrm{l,r}\}} \frac{|Q_s(\phi)|}{|Q|} H(Q_s(\phi)) \tag{6}$$

   where Shannon entropy $H(Q)$ is computed on the normalized histogram of body part labels $l_I(\mathbf{x})$ for all $(I,\mathbf{x}) \in Q$.
4. If the largest gain $G(\phi^\star)$ is sufficient, and the depth in the tree is below a maximum, then recurse for left and right subsets $Q_\mathrm{l}(\phi^\star)$ and $Q_\mathrm{r}(\phi^\star)$.

To keep the training times down we employ a distributed implementation. Training 3 trees to depth 20 from 1 million images takes about a day on a 1000 core cluster.

3.4 Joint Position Proposals

Body part recognition as described above infers per-pixel information. This information must now be pooled across pixels to generate reliable proposals for the positions of 3D skeletal joints. These proposals are the final output of our algorithm, and could be used by a tracking algorithm to self-initialize and recover from failure.

A simple option is to accumulate the global 3D centers of probability mass for each part, using the known calibrated depth. However, outlying pixels severely degrade the quality of such a global estimate. Instead we employ a local mode-finding approach based on mean shift [10] with a weighted Gaussian kernel.

We define a density estimator per body part as

$$f_c(\hat{\mathbf{x}}) \propto \sum_{i=1}^{N} w_{ic} \exp\left(- \left\| \frac{\hat{\mathbf{x}} - \hat{\mathbf{x}}_i}{b_c} \right\|^2 \right) , \tag{7}$$

where $\hat{\mathbf{x}}$ is a coordinate in 3D world space, N is the number of image pixels, w_{ic} is a pixel weighting, $\hat{\mathbf{x}}_i$ is the reprojection of image pixel $\mathbf{x}_i$ into world space given depth $d_I(\mathbf{x}_i)$, and b_c is a learned per-part bandwidth. The pixel weighting w_{ic} considers both the inferred body part probability at the pixel and the world surface area of the pixel:

$$w_{ic} = P(c|I, \mathbf{x}_i) \cdot d_I(\mathbf{x}_i)^2 . \tag{8}$$

This ensures density estimates are depth invariant and gave a small but significant improvement in joint prediction accuracy. Depending on the definition of body parts, the posterior $P(c|I, \mathbf{x})$ can be pre-accumulated over a small set of parts. For example, in our experiments the four body parts covering the head are merged to localize the head joint.

Mean shift is used to find modes in this density efficiently. All pixels above a learned probability threshold λ_c are used as starting points for part c. A final confidence estimate is given as a sum of the pixel weights reaching each mode. This proved more reliable than taking the modal density estimate.

The detected modes lie on the *surface* of the body. Each mode is therefore pushed back into the scene by a learned z offset ζ_c to produce a final joint position proposal. This simple, efficient approach works well in practice. The bandwidths b_c, probability threshold λ_c, and surface-to-interior z offset ζ_c are optimized per-part on a hold-out validation set of 5000 images by grid search. (As an indication, this resulted in mean bandwidth 0.065m, probability threshold 0.14, and z offset 0.039m).

4 Experiments

In this section we describe the experiments performed to evaluate our method. We show both qualitative and quantitative results on several challenging datasets, and compare with both nearest-neighbor approaches and the state of the art [13]. We provide further results in the supplementary material. Unless otherwise specified, parameters below were set as: 3 trees, 20 deep, 300k training images per tree, 2000 training example pixels per image, 2000 candidate features θ, and 50 candidate thresholds τ per feature.

Test data. We use challenging synthetic and real depth images to evaluate our approach. For our synthetic test set, we synthesize 5000 depth images, together with the ground truth body part labels and joint positions. The original mocap *poses* used to generate these images are held out from the training data. Our real test set consists of 8808 frames of real depth images over 15 different subjects, hand-labeled with dense body parts and 7 upper body joint positions. We also evaluate on the real depth data from [13]. The results suggest that effects seen on synthetic data are mirrored

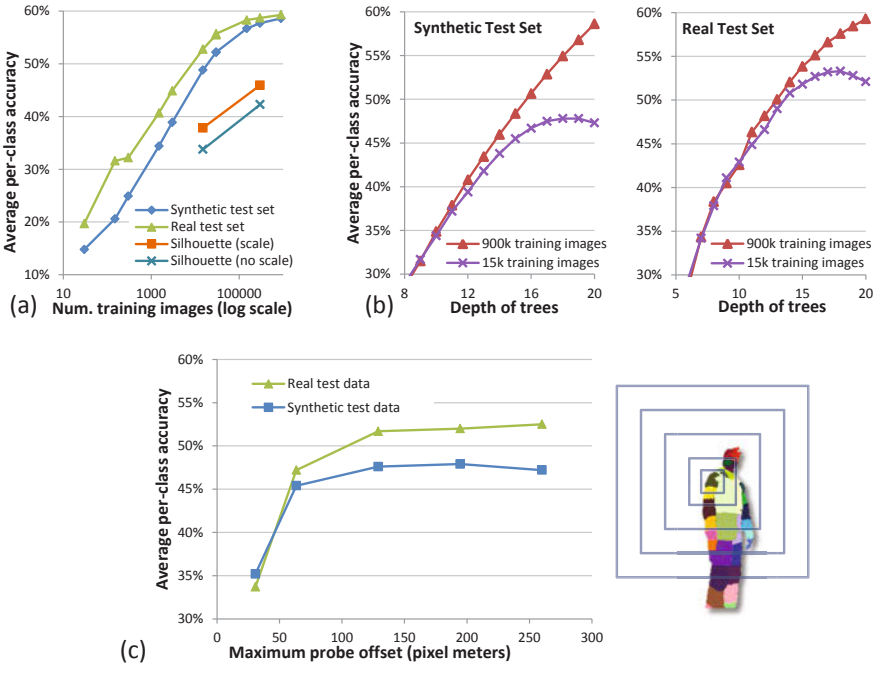


Fig. 6 Training parameters *vs.* classification accuracy. (a) Number of training images. (b) Depth of trees. (c) Maximum probe offset.

in the real data, and further that our synthetic test set is by far the 'hardest' due to the extreme variability in pose and body shape. For most experiments we limit the rotation of the user to $\pm 120°$ in both training and synthetic test data since the user is facing the camera (0°) in our main entertainment scenario, though we also evaluate the full 360° scenario.

Error metrics. We quantify both classification and joint prediction accuracy. For classification, we report the average per-class accuracy, *i.e.,* the average of the diagonal of the confusion matrix between the ground truth part label and the most likely inferred part label. This metric weights each body part equally despite their varying sizes, though mislabelings on the part boundaries reduce the absolute numbers.

For joint proposals, we generate recall-precision curves as a function of confidence threshold. We quantify accuracy as average precision per joint, or mean average precision (mAP) over all joints.The first joint proposal within D meters of the ground truth position is taken as a true positive, while other proposals also within D meters count as false positives. This penalizes multiple spurious detections near the correct position which might slow a downstream tracking algorithm. Any joint proposals outside D meters also count as false positives. Note that *all* proposals (not just the most confident) are counted in this metric. Joints invisible in the image are not penalized as false negatives. We set $D = 0.1$m below, approximately

the accuracy of the hand-labeled real test data ground truth. The strong correlation of classification and joint prediction accuracy (*c.f.* the blue curves in Figs. 6(a) and 8(a)) suggests the trends observed below for one also apply for the other.

4.1 Qualitative Results

Fig. 5 shows example inferences of our algorithm. Note high accuracy of both classification and joint prediction across large variations in body and camera pose, depth in scene, cropping, and body size and shape (*e.g.,* small child *vs.* heavy adult). The bottom row shows some failure modes of the body part classification. The first example shows a failure to distinguish subtle changes in the depth image such as the crossed arms. Often (as with the second and third failure examples) the most likely body part is incorrect, but there is still sufficient correct probability mass in distribution $P(c|I,\mathbf{x})$ that an accurate proposal can still be generated. The fourth example shows a failure to generalize well to an unseen pose, but the confidence gates bad proposals, maintaining high precision at the expense of recall.

Note that no temporal or kinematic constraints (other than those implicit in the training data) are used for any of our results. Despite this, per-frame results on video sequences in the supplementary material show almost every joint accurately predicted with remarkably little jitter.

4.2 Classification Accuracy

We investigate the effect of several training parameters on classification accuracy. The trends are highly correlated between the synthetic and real test sets, and the real test set appears consistently 'easier' than the synthetic test set, probably due to the less varied poses present.

Number of training images. In Fig. 6(a) we show how test accuracy increases approximately logarithmically with the number of randomly generated training images, though starts to tail off around 100k images. As shown below, this saturation is likely due to the limited model capacity of a 3 tree, 20 deep decision forest.

Silhouette images. We also show in Fig. 6(a) the quality of our approach on synthetic silhouette images, where the features in Eq. 1 are either given scale (as the mean depth) or not (a fixed constant depth). For the corresponding joint prediction using a 2D metric with a 10 pixel true positive threshold, we got 0.539 mAP with scale and 0.465 mAP without. While clearly a harder task due to depth ambiguities, these results suggest the applicability of our approach to other imaging modalities.

Depth of trees. Fig. 6(b) shows how the depth of trees affects test accuracy using either 15k or 900k images. Of all the training parameters, depth appears to have the most significant effect as it directly impacts the model capacity of the classifier. Using only 15k images we observe overfitting beginning around depth 17, but the enlarged 900k training set avoids this. The high accuracy gradient at depth 20

suggests even better results can be achieved by training still deeper trees, at a small extra run-time computational cost and a large extra memory penalty. Of practical interest is that, until about depth 10, the training set size matters little, suggesting an efficient training strategy.

Maximum probe offset. The range of depth probe offsets allowed during training has a large effect on accuracy. We show this in Fig. 6(c) for 5k training images, where 'maximum probe offset' means the max. absolute value proposed for both x and y coordinates of **u** and **v** in Eq. 1. The concentric boxes on the right show the 5 tested maximum offsets calibrated for a left shoulder pixel in that image; the largest offset covers almost all the body. (Recall that this maximum offset scales with world depth of the pixel). As the maximum probe offset is increased, the classifier is able to use more spatial context to make its decisions, though without enough data would eventually risk overfitting to this context. Accuracy increases with the maximum probe offset, though levels off around 129 pixel meters.

4.3 Joint Prediction Accuracy

In Fig. 7 we show average precision results on the synthetic test set, achieving 0.731 mAP. We compare an idealized setup that is given the *ground truth* body part labels to the real setup using inferred body parts. While we do pay a small penalty for using our intermediate body parts representation, for many joints the inferred results are both highly accurate and close to this upper bound. On the real test set, we have ground truth labels for head, shoulders, elbows, and hands. An mAP of 0.984 is achieved on those parts given the ground truth body part labels, while 0.914 mAP is achieved using the inferred body parts. As expected, these numbers are considerably higher on this easier test set.

Comparison with nearest neighbor. To highlight the need to treat pose recognition in *parts*, and to calibrate the difficulty of our test set for the reader, we compare with

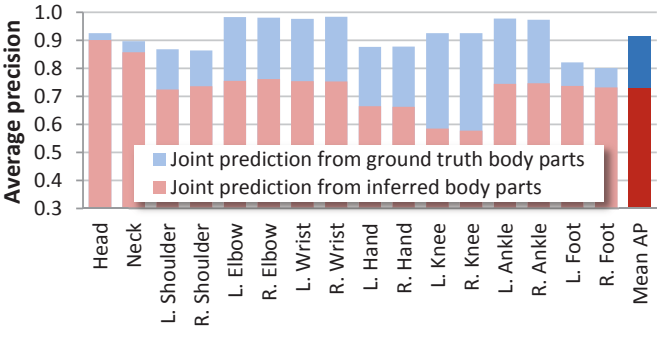


Fig. 7 Joint prediction accuracy. We compare the actual performance of our system (red) with the best achievable result (blue) given the ground truth body part labels.

two variants of exact nearest-neighbor whole-body matching in Fig. 8(a). The first, idealized, variant matches the ground truth *test skeleton* to a set of training exemplar skeletons with optimal rigid translational alignment in 3D world space. Of course, in practice one has no access to the test skeleton. As an example of a realizable system, the second variant uses chamfer matching [14] to compare the test image to the training exemplars. This is computed using depth edges and 12 orientation bins. To make the chamfer task easier, we throw out any cropped training or test images. We align images using the 3D center of mass, and found that further local rigid translation only reduced accuracy.

Our algorithm, recognizing in parts, generalizes better than even the idealized skeleton matching until about 150k training images are reached. As noted above, our results may get even better with deeper trees, but already we robustly infer 3D body joint positions and cope naturally with cropping and translation. The speed of nearest neighbor chamfer matching is also drastically slower (2 fps) than our algorithm. While hierarchical matching [14] is faster, one would still need a massive exemplar set to achieve comparable accuracy.

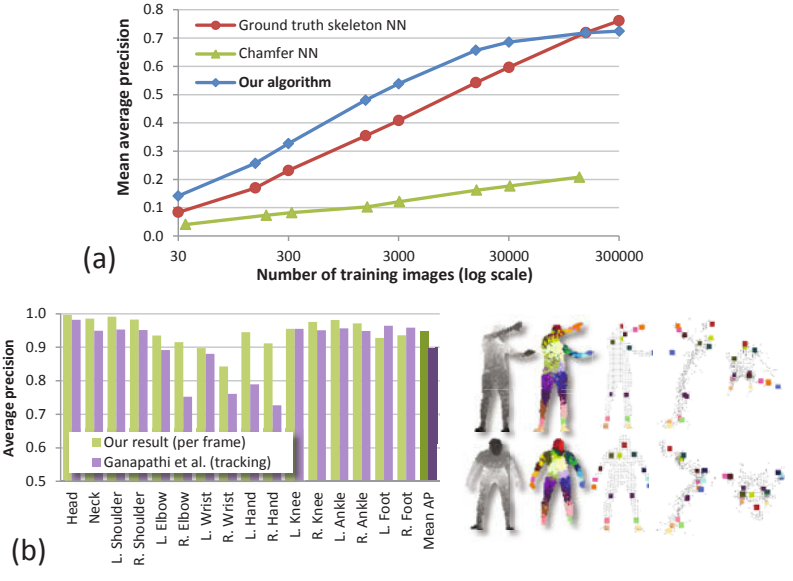


Fig. 8 Comparisons. (a) Comparison with nearest neighbor matching. (b) Comparison with [13]. Even without the kinematic and temporal constraints exploited by [13], our algorithm is able to more accurately localize body joints.

Comparison with [13]. The authors of [13] provided their test data and results for direct comparison. Their algorithm uses body part proposals from [28] and further tracks the skeleton with kinematic and temporal information. Their data comes from a time-of-flight depth camera with very different noise characteristics to our

structured light sensor. Without any changes to our training data or algorithm, Fig. 8(b) shows considerably improved joint prediction average precision. Our algorithm also runs at least 10x faster.

Full rotations and multiple people. To evaluate the full 360° rotation scenario, we trained a forest on 900k images containing full rotations and tested on 5k synthetic full rotation images (with held out poses). Despite the massive increase in left-right ambiguity, our system was still able to achieve an mAP of 0.655, indicating that our classifier can accurately learn the subtle visual cues that distinguish front and back facing poses. Residual left-right uncertainty after classification can naturally be propagated to a tracking algorithm through multiple hypotheses. Our approach can propose joint positions for multiple people in the image, since the per-pixel classifier generalizes well even without explicit training for this scenario. Results are given in Fig. 1 and the supplementary material.

Faster proposals. We also implemented a faster alternative approach to generating the proposals based on simple bottom-up clustering. Combined with body part classification, this runs at $\sim$ 200 fps on the Xbox GPU, *vs.* $\sim$ 50 fps using mean shift on a modern 8 core desktop CPU. Given the computational savings, the 0.677 mAP achieved on the synthetic test set compares favorably to the 0.731 mAP of the mean shift approach.

5 Discussion

We have seen how accurate proposals for the 3D locations of body joints can be estimated in super real-time from single depth images. We introduced body part recognition as an intermediate representation for human pose estimation. Using a highly varied synthetic training set allowed us to train very deep decision forests using simple depth-invariant features without overfitting, learning invariance to both pose and shape. Detecting modes in a density function gives the final set of confidence-weighted 3D joint proposals. Our results show high correlation between real and synthetic data, and between the intermediate classification and the final joint proposal accuracy. We have highlighted the importance of breaking the whole skeleton into parts, and show state of the art accuracy on a competitive test set.

As future work, we plan further study of the variability in the source mocap data, the properties of the generative model underlying the synthesis pipeline, and the particular part definitions. Whether a similarly efficient approach that can directly regress joint positions is also an open question. Perhaps a global estimate of latent variables such as coarse person orientation could be used to condition the body part inference and remove ambiguities in local pose estimates.

Acknowledgements. We thank the many skilled engineers in Xbox, particularly Robert Craig, Matt Bronder, Craig Peeper, Momin Al-Ghosien, and Ryan Geiss, who built the Kinect tracking system on top of this research. We also thank John Winn, Duncan Robertson,

Antonio Criminisi, Shahram Izadi, Ollie Williams, and Mihai Budiu for help and valuable discussions, and Varun Ganapathi and Christian Plagemann for providing their test data.

References

1. Agarwal, A., Triggs, B.: 3D human pose from silhouettes by relevance vector regression. In: Proc. CVPR (2004)
2. Amit, Y., Geman, D.: Shape quantization and recognition with randomized trees. Neural Computation 9(7), 1545–1588 (1997)
3. Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Ng, A.: Discriminative learning of markov random fields for segmentation of 3D scan data. In: Proc. CVPR (2005)
4. Autodesk MotionBuilder
5. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE Trans. PAMI 24 (2002)
6. Bourdev, L., Malik, J.: Poselets: Body part detectors trained using 3D human pose annotations. In: Proc. ICCV (2009)
7. Bregler, C., Malik, J.: Tracking people with twists and exponential maps. In: Proc. CVPR (1998)
8. Breiman, L.: Random forests. Mach. Learning 45(1), 5–32 (2001)
9. CMU Mocap Database, http://mocap.cs.cmu.edu/
10. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. PAMI 24(5) (2002)
11. Felzenszwalb, P.F., Huttenlocher, D.P.: Pictorial structures for object recognition. IJCV 61(1), 55–79 (2005)
12. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: Proc. CVPR (2003)
13. Ganapathi, V., Plagemann, C., Koller, D., Thrun, S.: Real time motion capture using a single time-of-flight camera. In: Proc. CVPR (2010)
14. Gavrila, D.M.: Pedestrian Detection from a Moving Vehicle. In: Vernon, D. (ed.) ECCV 2000. LNCS, vol. 1843, pp. 37–49. Springer, Heidelberg (2000)
15. Gonzalez, T.F.: Clustering to minimize the maximum intercluster distance. Theor. Comp. Sci. 38 (1985)
16. Grest, D., Woetzel, J., Koch, R.: Nonlinear Body Pose Estimation from Depth Images. In: Kropatsch, W.G., Sablatnig, R., Hanbury, A. (eds.) DAGM 2005. LNCS, vol. 3663, pp. 285–292. Springer, Heidelberg (2005)
17. Ioffe, S., Forsyth, D.: Probabilistic methods for finding people. IJCV 43(1), 45–68 (2001)
18. Kalogerakis, E., Hertzmann, A., Singh, K.: Learning 3D mesh segmentation and labeling. ACM Trans. Graphics 29(3) (2010)
19. Knoop, S., Vacek, S., Dillmann, R.: Sensor fusion for 3D human body tracking with an articulated 3D body model. In: Proc. ICRA (2006)
20. Lepetit, V., Lagger, P., Fua, P.: Randomized trees for real-time keypoint recognition. In: Proc. CVPR, vol. 2, pp. 775–781 (2005)
21. Microsoft Corp. Redmond WA. Kinect for Xbox 360
22. Moeslund, T.B., Hilton, A., Krüger, V.: A survey of advances in vision-based human motion capture and analysis. In: CVIU (2006)
23. Moosmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomized clustering forests. In: NIPS (2006)
24. Mori, G., Malik, J.: Estimating human body configurations using shape context matching. In: Proc. ICCV (2003)
25. Navaratnam, R., Fitzgibbon, A.W., Cipolla, R.: The joint manifold model for semi-supervised multi-valued regression. In: Proc. ICCV (2007)

26. Ning, H., Xu, W., Gong, Y., Huang, T.S.: Discriminative learning of visual words for 3D human pose estimation. In: Proc. CVPR (2008)
27. Okada, R., Soatto, S.: Relevant Feature Selection for Human Pose Estimation and Localization in Cluttered Images. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 434–445. Springer, Heidelberg (2008)
28. Plagemann, C., Ganapathi, V., Koller, D., Thrun, S.: Real-time identification and localization of body parts from depth images. In: Proc. ICRA (2010)
29. Poppe, R.: Vision-based human motion analysis: An overview. CVIU 108 (2007)
30. Quinlan, J.R.: Induction of decision trees. Mach. Learn. (1986)
31. Ramanan, D., Forsyth, D.A.: Finding and tracking people from the bottom up. In: Proc. CVPR (2003)
32. Rogez, G., Rihan, J., Ramalingam, S., Orrite, C., Torr, P.H.S.: Randomized trees for human pose detection. In: Proc. CVPR (2008)
33. Shakhnarovich, G., Viola, P., Darrell, T.: Fast pose estimation with parameter sensitive hashing. In: Proc. ICCV (2003)
34. Sharp, T.: Implementing Decision Trees and Forests on a GPU. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 595–608. Springer, Heidelberg (2008)
35. Shepherd, B.A.: An appraisal of a decision tree approach to image classification. In: IJCAI (1983)
36. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image categorization and segmentation. In: Proc. CVPR (2008)
37. Siddiqui, M., Medioni, G.: Human pose estimation from a single view point, real-time range sensor. In: CVCG at CVPR (2010)
38. Sidenbladh, H., Black, M.J., Sigal, L.: Implicit Probabilistic Models of Human Motion for Synthesis and Tracking. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2350, pp. 784–800. Springer, Heidelberg (2002)
39. Sigal, L., Bhatia, S., Roth, S., Black, M.J., Isard, M.: Tracking loose-limbed people. In: Proc. CVPR (2004)
40. Tu, Z.: Auto-context and its application to high-level vision tasks. In: Proc. CVPR (2008)
41. Urtasun, R., Darrell, T.: Local probabilistic regression for activity-independent human pose inference. In: Proc. CVPR (2008)
42. Wang, R.Y., Popović, J.: Real-time hand-tracking with a color glove. In: Proc. ACM SIGGRAPH (2009)
43. Winn, J., Shotton, J.: The layout consistent random field for recognizing and segmenting partially occluded objects. In: Proc. CVPR (2006)
44. Zhu, Y., Fujimura, K.: Constrained Optimization for Human Pose Estimation from Depth Sequences. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) ACCV 2007, Part I. LNCS, vol. 4843, pp. 408–418. Springer, Heidelberg (2007)

Scale-Invariant Vote-Based 3D Recognition and Registration from Point Clouds

Minh-Tri Pham, Oliver J. Woodford, Frank Perbet, Atsuto Maki,
Riccardo Gherardi, Björn Stenger, and Roberto Cipolla

Abstract. This chapter presents a method for vote-based 3D shape recognition and registration, in particular using mean shift on 3D pose votes in the space of direct similarity transformations for the first time. We introduce a new distance between poses in this space—the SRT distance. It is left-invariant, unlike Euclidean distance, and has a unique, closed-form mean, in contrast to Riemannian distance, so is fast to compute. We demonstrate improved performance over the state of the art in both recognition and registration on a (real and) challenging dataset, by comparing our distance with others in a mean shift framework, as well as with the commonly used Hough voting approach.

1 Introduction

This chapter concerns itself with vote-based pose estimation techniques. These arise in many vision tasks including 2D object detection [21, 28, 37], motion segmentation [39, 40], and 3D shape registration and recognition [11, 20, 41]. These methods all share a common two stage framework: First they generate an empirical distribution of pose through the collation of a set of possible poses, or *votes*. The votes are often computed by matching local features from a test object to those in a library with known pose [11, 20, 21, 28, 37, 39, 40, 41], or by learning a function that maps features to votes [16, 29]. The second step is then to find one or more "best" poses in the distribution (the maxima, in the case of ML/MAP estimation). This curation of

Minh-Tri Pham · Oliver J. Woodford · Frank Perbet · Atsuto Maki · Riccardo Gherardi · Björn Stenger
Toshiba Research Europe Ltd
e-mail: firstname.lastname@crl.toshiba.co.uk

Roberto Cipolla
Department of Engineering, University of Cambridge, UK
e-mail: cipolla@eng.cam.ac.uk

R. Cipolla et al. (Eds.): Machine Learning for Computer Vision, SCI 411, pp. 137–162.
springerlink.com

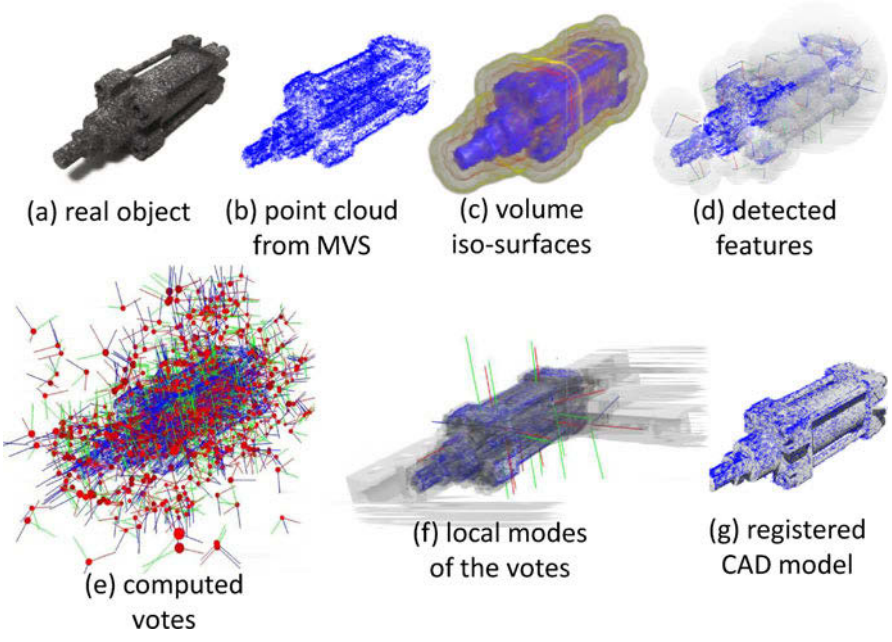


Fig. 1 Our System. for 3D-shape-based object recognition and registration. (a) Real object, fabricated from a CAD model. (b) Point cloud extracted using a multi-view stereo (MVS) system. (c) Iso-surfaces of the scalar volume computed from the points. (d) Features (with full scale, rotation and translation pose) detected in the volume. (e) Votes for the object centre, based on detected features matched with a library of learnt features. (f) Local modes of the votes. (g) The registered CAD model.

data prior to inference makes such vote-based approaches more efficient and robust than competing techniques, *e.g.*, global or appearance-based methods [26].

Two compelling methods in finding best poses are Hough voting and mean shift. In Hough voting the standard approach is to compute the probabilities on a regular grid over the pose parameter space. This discretization leads to loss of accuracy, as well as a complexity exponential in the pose dimensionality, but ensures coverage of the entire space. Mean shift [9] iteratively finds local maxima of probability, resulting in initialization issues but also high accuracy. The complexity of an iteration is usually linear in the pose dimensionality. The two methods are therefore somewhat complementary; indeed they are often used together [21, 28].

While Hough voting can be easily applied to any space (in our case that of all poses), this is not straightforward for mean shift; each iteration requires the computation of a weighted average of input votes, formulated as a least squares minimization of distances from input votes to the mean. In Euclidean space this minimization yields a unique, closed-form solution—the arithmetic mean. When poses lie on a non-linear manifold this mean is typically outside the manifold, requiring a projection onto it. A more direct approach is to minimize the geodesic arclengths over the manifold, known as the Riemannian distance.

requiring a projection onto it. A more direct approach is to minimize the geodesic arclengths over the manifold, known as the Riemannian distance.

In this chapter we focus on 3D shape recognition and registration, as part of a system (see Fig. 1) for recognizing industrial parts. However, unlike existing approaches, where objects of interest are of either fixed (or omitted) scale [40] or rotation [21, 28, 37], here we recognize and register objects in the direct similarity group: the group of isotropic similarity transformations parameterized by translation, rotation *and* scale [36]. Scale is necessary when the input data's scale is unknown, or when there is high intra-class scale variation. Rotation is necessary for full registration, leading to more accurate recognition. The resulting 7D pose space is currently too large to apply Hough voting to in practice [19]. Here we use mean shift, for which scale and rotation also introduce problems using existing distances: Euclidean distance is scale variant, and the induced mean of poses has a bias in scale. The mean of poses using Riemannian distance has no closed-form solution, even when the poses are pure rotations [25], and is slow to compute [38].

The contribution of this work is to introduce a new distance on the direct similarity group. The distance provides scale, rotation and translation-invariance concomitantly. The weighted mean of this distance is unique, closed-form, and fast to compute, as well as having several key properties discussed in Sect. 2.4.3. We demonstrate the distance's performance in mean shift, in the context of our 3D shape registration and recognition system, comparing it with other distances on the same space, as well as a Hough voting method.

The chapter is laid out as follows: The next section reviews the literature relevant to 3D shape recognition and registration inference, as well as how our method is positioned compared to existing approaches. In the following section, we introduce our new distance on the direct similarity group, and its associated mean. In the final two sections, we present our experiments, before concluding.

2 Background

We start with discussing two main trends in the literature: global approaches versus local approaches, in which our method belongs to the latter. We then review how a local appearance model is learned and used for generating votes from features in local approaches. Our method extracts features using the standard Difference-of-Gaussian (DoG) operator and matches features between the scene and the model to generate votes. In the last part of the section, we review the inference techniques used for vote-based pose estimation, and take a closer look at mean shift applied to this task.

2.1 Global Approaches vs. Local Approaches

Recognizing and registering rigid objects from 3D point clouds is a well-known problem in computer vision [6, 22, 23]. Often, the 3D point clouds obtained by

different sensors such as laser scans, time-of-flight cameras, or stereo systems [43] contain small, irrelevant, neighbouring clutter in addition to the relevant data coming from the objects.In most cases, the relevant data themselves do not capture full shapes. Two main approaches to solve the problem are: global approaches and local approaches. Global approaches recognize objects by relying on global features, *i.e.*, features extracted from the complete 3D geometry of the point cloud. Examples include spherical harmonics [18, 35], shape moments [35], and shape histograms [29]. It is difficult to handle partial shapes using these approaches, since global features are sensitive to both absence of shape parts and occurrence of clutter.

Recent works in 2D object detection and object class categorization [21, 28] have shown the advantage of using local, rather than global, features in dealing with occlusions and clutter. In 3D, similar success stories have been reported with methods using local features [8, 15, 17, 20, 24, 41]. These approaches can integrate information from a large number of object parts. They demonstrate good generalization as they are free to combine parts observed on different training examples. Spin Images by Johnson and Hebert [17] is arguably the most popular early work, in which local 3D descriptions are represented as 2D histograms of points falling within a cylindrical region by means of a plane that "spins" around the normal, and recognition is done by matching spin images, grouping similar matches and verifying each output pose. Many local features and descriptors have been proposed thereafter, with new ones being more discriminative, more repeatable, more robust to noise and more invariant to local rigid transformations. Chen and Bhanu [8] compute histograms of normals and shape indices for describing local regions. The 3D Shape Context of Frome *etal*. [15] extends Spin Images' basic idea to computing 3D histograms of points within a sphere around a feature point. Mian *etal*. [24] accumulate 3D histograms of mesh triangles within a cubic support. Rusu *etal*. [34] propose Point Feature Histograms describing the local geometry of a point and its k nearest neighbours. Knopp *etal*. [20] extend the SURF descriptor from 2D to 3D and show how 3D shape recognition can be improved by a Hough-transform based approach. Petrelli and Di Stefano [33] improve the repeatability of local reference frames via point normals. Surveys of local features and descriptors 3D methods are available in [6, 22, 23].

Many of these approaches share a common vote-based framework. They first learn a local appearance model for the object classes to be recognized and registered, which maps, either directly or indirectly, features to ground truth object identities and poses. During inference, features from the scene point cloud are extracted. Via the local appearance model, each of these features generates one or more votes, representing hypotheses that an object of a given pose exists. The votes can be viewed as points of a kernel density estimator that estimates the joint probability density function of both object identity and pose. Local modes of the kernel density function are found via a suitable mode-seeking approach, and returned as final object identities and poses. Methods such as Iterative Closest Point [5] and variants, are able to further refine the output poses, if necessary.

Following these approaches, we use the standard vote-based framework in our 3D recognition and registration. However, our approach differs from existing

approaches in that we infer simultaneously scale, rotation and translation in 3D. Due to the introduction of scale, the pose space becomes too large (7D) for existing Hough transform-based approaches to work with, while existing mode-seeking methods like mean shift have bias in scale, as to be seen in the following sections.

2.2 Learning a Local Appearance Model

2.2.1 Feature Extraction

Salient interest regions are extracted over location and scale from a variety of point cloud instances of objects of the same class (in our case 20 point cloud instances per class) using 3D interest point detectors like 3D SURF [20]. For each interest region, *i.e.*, a sphere centered at $\mathbf{c}$ with radius r, a 3D canonical orientation based on the geometry of the points inside the region is computed, for example by finding the principal directions of the points using PCA [24], or by fitting a local surface and then finding most repeatable directions on the surface from the center of the sphere [33]. A local reference frame is created as a result, hereinafter *feature frame*, originating at $\mathbf{c}$, with one unit length equal to r, and with 3D orientation coinciding with the 3D canonical orientation. The feature frame is specified uniquely by a 3D direct similarity transformation $\mathbf{F} \in S^+(3)$ [36],

$$\mathbf{F} = \begin{bmatrix} s(\mathbf{F})\mathbf{R}(\mathbf{F}) & \mathbf{t}(\mathbf{F}) \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}, \tag{1}$$

which converts the coordinates of a 3D point from the global coordinate system to the feature frame. Here, $s(\mathbf{F}) \in \mathbb{R}^+$, $\mathbf{R}(\mathbf{F}) \in SO(3,\mathbb{R})$, and $\mathbf{t}(\mathbf{F}) \in \mathbb{R}^3$ specify the scale, rotation, and translation components of $\mathbf{F}$, respectively. A low-dimensional feature descriptor $\mathbf{d} \in \mathbb{R}^k$ (for some positive integer number k) is extracted based on the distribution of the coordinates of the points inside the region with respect to $\mathbf{F}$.

Each point cloud instance is associated with a local reference frame specifying the ground truth pose of the object captured in the point cloud, hereinafter *object frame*. Unlike most existing 3D approaches where an object pose specifies translation only [41], translation and scale [20], or translation and rotation [13], in our system an object pose specifies scale and rotation and translation altogether, hence dealing with a larger pose space than existing works. Here we treat scale as part of an object pose and choose an object frame originating at the center location of the object, with 3D orientation the same as the object's orientation, and with one unit length equal to the object scale. Analogously to the feature frames, an object frame is specified uniquely by 3D direct similarity transformation $\mathbf{X} \in S^+(3)$,

$$\mathbf{X} = \begin{bmatrix} s(\mathbf{X})\mathbf{R}(\mathbf{X}) & \mathbf{t}(\mathbf{X}) \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}, \tag{2}$$

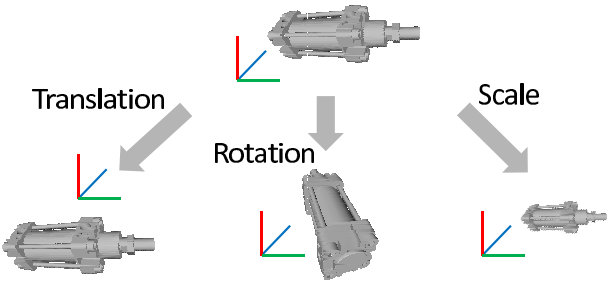


Fig. 2 Effects of translation, rotation, and scale in transforming the pose of an object.

converting the coordinates of a 3D point from the global coordinate system to the object frame. Since object frame and object pose are equivalent terms, from here onwards they are used interchangeably.

An training feature consisting of a feature descriptor $\mathbf{d}$, an object class identity $j \in \{1,..,J\}$ (where J is the number of classes), and a feature-to-object transformation $\mathbf{T} = \mathbf{X}\mathbf{F}^{-1}$ is formed for each detected interest region. Note that although both matrices $\mathbf{F}$ and $\mathbf{X}$ are computed from the global coordinate system, since the feature frame is covariant to the object pose, the resultant feature-to-object transformation $\mathbf{T}$ solely depends on the shape of the object. In other words, it is pose-invariant.

The collection of all training features extracted from every object class, denoted as $e_m = (\mathbf{d}_m, j_m, \mathbf{T}_m = \mathbf{X}_m\mathbf{F}_m^{-1})$ for $m \in \{1,..,M\}$ where M is the number of training features, can be viewed as the local appearance model of the object classes.

2.2.2 Learning the Feature-to-Vote Mapping

Existing approaches differ in how the votes are generated from features extracted from the scene, hereinafter *scene feature*s. There are three main approaches: (1) direct matching of scene features with training features, (2) unsupervised clustering of training features into visual words followed by matching of scene features with visual words, and (3) supervised learning to directly map each scene feature to one or more votes.

In approaches that match scene features with training features directly, all the training features are kept as a library of exemplars. Hence, the cost for training is very low. During inference, each scene feature $f = (\mathbf{d}^{'}, \mathbf{F}^{'})$ (with descriptor $\mathbf{d}^{'}$ and feature frame $\mathbf{F}^{'}$) is matched with every exemplar in the library and each match generates a vote. In Tombari and Di Stefano's work [41], matching of f with an exemplar e_m is done by thresholding the Euclidean distance $\|\mathbf{d}_m - \mathbf{d}'\|$ with a predefined threshold ε. If it is a match, *i.e.*, $\|\mathbf{d}_m - \mathbf{d}'\| < \varepsilon$, a vote for an object of class j_m at pose $\mathbf{T}_m\mathbf{F}^{'}$ is generated. Drost *etal*. [11] use hashing to match a feature descriptor with model descriptors instead. A vote may optionally have a weight to reflect the relative matching score and other prior probabilities, as shown in, for instance, the work of Knopp *etal*. [20]. It makes sense to use $\mathbf{T}_m\mathbf{F}^{'}$ to predict the object pose,

since if we transform the object specified by e_m, albeit unknown, so that the feature used for the construction of e_m aligns perfectly with f, *i.e.*, $\mathbf{F}_m = \mathbf{F}'$, the transformed object pose must be $\mathbf{T}_m\mathbf{F}'$.

Since matching every scene feature with every training feature is a time-costly process, it may be beneficial to group training features of similar descriptors coming from the same class into a visual word using an unsupervised clustering approach [20, 21]. Such a strategy would reduce the number of exemplars in the library, hence increasing the matching efficiency. However, the sizes of the clusters must be chosen carefully, or the false positive rate may increase [21, 41].

In 2D object recognition, the idea of grouping training features of similar appearances is advanced further, by using discriminative and supervised clustering rather than unsupervised clustering, allowing one to optimize the visual words to produce more reliable votes in the vote space. Gall and Lempitsky [16] train a Hough forest that maps a feature directly to multiple votes. However, each node of their Hough tree is trained to either minimize the class uncertainty or the pose uncertainty. Okada [27] instead introduces a combined objective function for training a node. Both methods have shown significant improvements over the unsupervised approach of Leibe *et al*. [21].

It would be tempting to apply this idea to 3D. However, an immediate challenge is how to model uncertainty of a set of 3D poses. In 2D, Gall and Lempitsky work with 2D center points, and Okada works with 2D points plus scale, the variance of which is sufficient to model the uncertainty. In our case, the existence of both 3D rotation and scale makes the pose space a non-linear manifold. Any uncertainty measurement based on the notion of Euclidean distance, including variance, would have a bias in scale, as to be discussed in Sect. 2.4.1.

In our approach, we use a standard feature extraction process, as described in Sect. 4. Similar to Tombari and Di Stefano [41], we use the dataset of training features as the local appearance model without clustering them into visual words. As the scope of this work is to introduce a distance that is efficient and more importantly, unbiased by scale, the task of modeling pose uncertainty, and subsequently supervised learning of visual words, is left for future work.

2.3 Finding Local Modes in the Vote Space

The inference stage involves the computation of local maxima of the joint kernel density function $p(j,\mathbf{X})$ of object identity $j \in \{1,..,J\}$ and pose $\mathbf{X} \in S^+(3)$ represented by a set of (weighted) votes generated from an input point cloud. Since j is a discrete variable, we can search for local maxima in each of the J conditional distributions $p(\mathbf{X}|j)$ instead. Without loss of generality, let us assume the form of $p(\mathbf{X}|j)$ as:

$$p(\mathbf{X}|j) = \sum_{i=1}^{N_j} \lambda_i H(\mathbf{X}, \mathbf{X}_i) \tag{3}$$

Table 1 Methods of pose estimation over different transformations. **t**: translation; **R**: rotation; s: scale. *Indicates 2D space.

	t	**t**, s	**t**, **R**	**t**, **R**, s
Hough	[41]	[20]	[13]	[19]
Mean shift	–	[21, 28, 37]*	[40]	[39]*, **This work**

where N_j denotes the number of votes for class j, $H(\cdot,\cdot)$ denotes a kernel function, and with respect to the i^{th} vote for class j, $\lambda_i \geq 0$ denotes the weight and $\mathbf{X}_i \in S^+(3)$ denotes the predicted pose. Here, the weights are normalized, *i.e.*, $\sum_i \lambda_i = 1$, so that $p(\mathbf{X}|j)$ is a proper probability density function. Although we are concerned with 3D transformations, the discussion in the remainder of the chapter assumes n-dimensional transformations for an arbitrary $n > 0$.

Two main techniques for finding modes of $p(\mathbf{X}|j)$ are Hough voting (an extension of the Generalized Hough Transform [4]) and mean shift [9].

In Hough voting, the input space is partitioned into a finite number of L bins, *i.e.*, $S^+(n) = \bigcup_{l=1}^{L} B_l$ where $B_i \cap B_j = \emptyset$ for all $i \neq j$. For each bin B_l, the weights of the votes with poses belonging to B_l are summed up. Modes are found by returning bins with largest sums of weights. Using Hough voting, Khoshelham [19] quantizes the 7D space of 3D translation, rotation and scale for object registration. This creates a trade-off between pose accuracy and computational requirements, the latter proving to be costly. Other methods seek to reduce this complexity by shrinking the pose space and marginalizing over some parameters. Fisher *etal*. [13] quantize translations and rotations in two separate 3D arrays; peak entries in both arrays indicate the pose of the object, but multiple objects create ambiguities. Knopp *etal*. [20] show effective object recognition using Hough voting over 3D translation and scale. Tombari and Di Stefano [41] first compute Hough votes over translation, assuming known scale in their 3D object recognition and registration application, then determine rotation by averaging the rotations at each mode. Geometric hashing [11, 24] is a similar technique to Hough voting which reparameterizes pose in a lower dimensional space before clustering. However, all these dimensionality reduction techniques lead to an increased chance of false positive detections. Another issue with Hough voting is that it returns bins, not poses, as output. One still needs a way to select the best pose, or to compute a representative pose, for each output bin.

Mean shift avoids the trade-off suffered by Hough voting methods, being both accurate and having lower (usually[1] linear) complexity in the pose dimensionality, making it suitable for inference in the full 7D pose space of the direct similarity group in 3D. To date it has been used in 2D applications: object detection over translation and scale [21, 28, 37], and motion segmentation over affine transformations [40], as well as in 3D for motion segmentation over translation and rotation [39]. Mean shift relies on a kernel function typically in the form,

[1] Certain distance computations are not linear, *e.g.*, that of Sect. 2.4.2.

$$H(\mathbf{X},\mathbf{X}_i) = \frac{1}{\zeta} K(d^2(\mathbf{X},\mathbf{X}_i)), \tag{4}$$

where $d(\cdot,\cdot)$ is a distance function and $K(\cdot)$ is a non-negative non-increasing univariate function, and ζ is a normalization factor so that $H(\cdot,\mathbf{X}_i)$ is a proper probability density function. Choosing the distance function $d(\cdot,\cdot)$, is crucial for mean shift as it directly changes the locations and the number of the output modes. On non-Euclidean spaces, even the Euclidean distance yields undesired behaviours as to be shown in Sect. 2.4. This is the first contribution we know of to apply mean shift to a 3D application using translation, rotation and scale simultaneously. A reason this has not done before could be the problems associated with computing means using existing Euclidean and Riemannian distances in the direct similarity group $S^+(n)$. We now review mean shift in more details and discuss distance functions in this space. In what follows, we omit index j since it is clear from the context that j is given.

Algorithm 1. Mean shift [9] (for notation see text)

Require: $\mathcal{X} = \{\mathbf{X}_i, \lambda_i\}_{i=1}^N$, distance function $d(\cdot,\cdot)$
1: Initialize $\mathbf{X}$
2: **repeat**
3: $\quad \mathbf{X}_{\text{old}} := \mathbf{X}$
4: $\quad w_i := \lambda_i K(d^2(\mathbf{X},\mathbf{X}_i)) \qquad \forall i = 1,..,N$
5: $\quad \mathbf{X} := \arg\min_{\mathbf{X}} \sum_i w_i d^2(\mathbf{X},\mathbf{X}_i)$
6: **until** $d(\mathbf{X}_{\text{old}},\mathbf{X}) < \varepsilon$
7: **return** $\mathbf{X}$

2.4 Mean Shift

The mean shift algorithm [9] (Algorithm 1) is a popular algorithm for finding local modes by coordinate ascent in kernel density estimation. Given a distance function $d(\cdot,\cdot)$ on the input space, the kernel density estimator is given by

$$\hat{f}_K(\mathbf{X}) = \sum_{i=1}^{N} \frac{1}{\zeta} \lambda_i K(d^2(\mathbf{X},\mathbf{X}_i)), \tag{5}$$

where $\mathbf{X}$ is the random variable, $\mathcal{X} = \{\mathbf{X}_i, \lambda_i\}_{i=1}^N$ is a set of input points with weights $\lambda_i \geq 0$, $K(\cdot) \geq 0$ is a kernel function, and ζ is a volume density function which normalizes $K(d^2(\cdot,\mathbf{X}_i))$. The most common (and our) choice for $K(\cdot)$ is the Gaussian kernel, $\exp\left(-\frac{\cdot}{2\sigma^2}\right)$, where σ is the bandwidth of the kernel. On Euclidean spaces a natural choice for $d()$ is the Euclidean distance, $d_{\text{E}}()$; *e.g.*, if $\mathbf{X}$ and $\mathbf{Y}$ are matrices, $d_{\text{E}}(\mathbf{X},\mathbf{Y}) = \|\mathbf{X}-\mathbf{Y}\|_{\text{F}}$ where $\|\cdot\|_{\text{F}}$ is the Frobenius norm. Under the Euclidean distance, the solution of step 5 in Algorithm 1

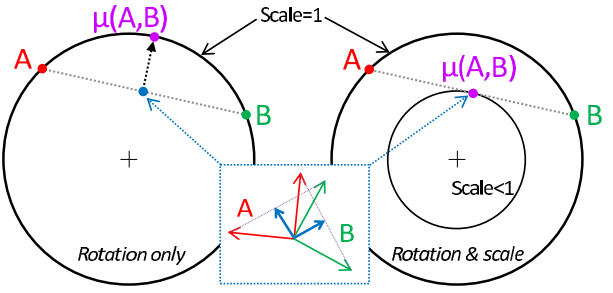


Fig. 3 Scale bias of the extrinsic mean. Let us consider $S^+(2)$ (without translation): on a plane, a rotation can be represented as a point on a circle, the radius being the scale. *Left*: with rotation only, the arithmetic mean of **A** and **B** leads to a smaller scale but the reprojection onto the manifold (*i.e.*, the unit circle) gives a reasonable result. *Right*: with rotation and scale, the mean is already on the manifold, but with a smaller scale.

$$\mu(\mathcal{X}) = \arg\min_{\mathbf{X}} \sum_i w_i d^2(\mathbf{X}, \mathbf{X}_i), \tag{6}$$

also known in the literature as a Fréchet mean [14], becomes an arithmetic mean, *i.e.*, $\mu(\mathcal{X}) = \frac{\sum_i w_i \mathbf{X}_i}{\sum_i w_i}$.

In pose estimation, votes are represented by linear transformations which form a matrix Lie group. This chapter is concerned with the direct similarity group $S^+(n) \subset GL(n+1,\mathbb{R})$, which is the set of all affine transformation matrices $\mathbf{X} \in S^+(n)$ acting on $\mathbb{R}^n$ preserving angles and orientations [36]. When applying mean shift on a matrix Lie group, the choice of $d()$ is crucial since it affects both the computation of weights and the mean (steps 4 & 5 of Algorithm 1). Two well-known distances arise in the literature: Euclidean and Riemannian. We now review how existing methods utilize these distances in mean shift on matrix Lie groups.

2.4.1 Euclidean Distance

Given a matrix Lie group $\mathcal{G} \subset GL(n,\mathbb{R})$, since $GL(n,\mathbb{R}) \subset \mathbb{R}^{n^2}$ (up to an isomorphism), the most straightforward way to apply mean shift on $\mathcal{G}$ is to run Euclidean mean shift on $\mathbb{R}^{n^2}$ instead. However, at each iteration the arithmetic mean may not lie in $\mathcal{G}$. It is therefore projected back to $\mathcal{G}$ via the mapping:

$$\pi : \mathbb{R}^{n^2} \to \mathcal{G} : \pi(\mathbf{X}) = \arg\min_{\mathbf{Y} \in \mathcal{G}} \|\mathbf{Y} - \mathbf{X}\|_{\mathrm{F}}^2. \tag{7}$$

The projected arithmetic mean, $\mu(\mathcal{X}) = \pi\left(\frac{\sum_i w_i \mathbf{X}_i}{\sum_i w_i}\right)$, is referred to in the literature as the *extrinsic* mean [25, 38].

Mean shift using Euclidean distance (extrinsic mean shift) has shown good results on Stiefel and Grassmann manifolds [7]. However, there are two drawbacks with extrinsic mean shift applied to $S^+(n)$. First, $d_{\mathrm{E}}()$ is invariant to rotation and translation but not to scaling, making the weights, w_i, computed by mean shift scale

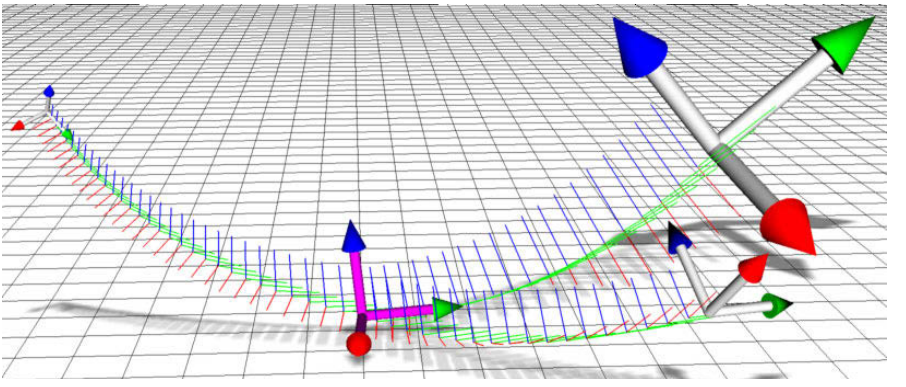


Fig. 4 The intrinsic mean. Three poses in $S^+(3)$ (with different scales, rotations and translations) and their intrinsic mean (pink). The geodesics between the mean and input poses are also drawn. Note that the shortest distance between two transformations is not necessarily a straight line in terms of translation.

variant. Thus, although the extrinsic mean is scale-covariant[2], extrinsic mean shift is *not*. Second, the extrinsic mean of rotation and scale transformations causes a bias towards smaller scales, as illustrated in Fig. 3.

2.4.2 Riemannian Distance

An alternative choice for $d()$ is the Riemannian distance. $d_{\mathrm{R}}()$. Given $\mathbf{X},\mathbf{Y} \in S^+(n)$, $d_{\mathrm{R}}(\mathbf{X},\mathbf{Y})$ is defined as the arclength of the geodesic between $\mathbf{X}$ and $\mathbf{Y}$, *i.e.*, the shortest curve along the manifold connecting $\mathbf{X}$ and $\mathbf{Y}$ (see Fig. 4). In general $d_{\mathrm{R}}(\mathbf{X},\mathbf{Y})$ is difficult to compute, but if $\mathbf{Y}$ is located within the open neighbourhood bounded by the cut locus of $\mathbf{X}$ in $S^+(n)$ (defined in [31]) then $d_{\mathrm{R}}(\mathbf{X},\mathbf{Y}) = \left\|\mathrm{logm}\left(\mathbf{X}^{-1}\mathbf{Y}\right)\right\|_{\mathrm{F}}$, where $\mathrm{logm}(\cdot)$ is the matrix logarithm. This requirement is not too restrictive in practice; in $S^+(n)$ the rotation angle should not reach π radians [45]. For example, in $SO(2,\mathbb{R})$ the cut locus of $\mathbf{X}$ is just a single point: $-\mathbf{X}$ [25].

Since $d_{\mathrm{R}}()$ depends only on the *intrinsic* geometry of $\mathcal{G}$, the Fréchet mean (*i.e.*, mean defined as the solution of Eq. (6)) using $d_{\mathrm{R}}()$ is called the intrinsic mean [25, 31]. Efficient formulations of $d_{\mathrm{R}}()$ exist for some $\mathcal{G}$, notably $SE(3)$ [2], which can be adapted to $S^+(3)$. However, in $S^+(n)$ for $n>3$, $d_{\mathrm{R}}()$ generally has no efficient formulation, taking $O(n^4)$ time to compute [10].

Intrinsic mean shift methods have been proposed [7, 40]. The intrinsic mean itself has multiple non-closed-form solutions [25]; in our experiments we compute an approximation using a single step[3] of the iterative method of [40].

[2] Scale-covariant means a scale transformation of input data produces the same transformation on the output.

[3] This is equivalent to computing a mean using the *log-Euclidean* distance [3], $d(\mathbf{X},\mathbf{Y}) = \|\mathrm{logm}(\mathbf{X}) - \mathrm{logm}(\mathbf{Y})\|_{\mathrm{F}}$.

2.4.3 Properties of a Good Distance in $S^+(n)$

In the context of mean shift, and subsequent to our overview of Euclidean and Riemannian distances, we propose the following list of desirable properties for a distance in $S^+(n)$ and its associated mean:

1. *Unique:* The mean should have a unique solution.
2. *Closed-form:* For efficient computation, the mean should have a closed-form solution.
3. *Scale-compatible:* If all rotations and translations are equal, the mean should behave as an average of the scales. Mathematically, if $\forall \mathbf{X}_i \in \mathcal{X} : \mathbf{R}(\mathbf{X}_i) = \mathbf{R}', \mathbf{t}(\mathbf{X}_i) = \mathbf{t}'$ for some $\mathbf{R}'$ and $\mathbf{t}'$, then we would like $\mathbf{R}(\mu(\mathcal{X})) = \mathbf{R}'$, $\mathbf{t}(\mu(\mathcal{X})) = \mathbf{t}'$, and $s(\mu(\mathcal{X}))$ to be an average of $s(\mathbf{X}_i)$'s. In this case, we say that μ is scale-compatible.
4. *Rotation-compatible:* If $\forall \mathbf{X}_i \in \mathcal{X} : s(\mathbf{X}_i) = s'$, $\mathbf{t}(\mathbf{X}_i) = \mathbf{t}'$, then $s(\mu(\mathcal{X})) = s'$, $\mathbf{t}(\mu(\mathcal{X})) = \mathbf{t}'$ and $\mathbf{R}(\mu(\mathcal{X}))$ is an average of $\mathbf{R}(\mathbf{X}_i)$'s.
5. *Translation-compatible:* If $\forall \mathbf{X}_i \in \mathcal{X} : s(\mathbf{X}_i) = s'$, $\mathbf{R}(\mathbf{X}_i) = \mathbf{R}'$, then $s(\mu(\mathcal{X})) = s'$, $\mathbf{R}(\mu(\mathcal{X})) = \mathbf{R}'$ and $\mathbf{t}(\mu(\mathcal{X}))$ is an average of $\mathbf{t}(\mathbf{X}_i)$'s.
6. *Left-invariant:* A left-invariant distance is one that is unchanged by any post-transformation, *i.e.*, $d(\mathbf{ZX}, \mathbf{ZY}) = d(\mathbf{X}, \mathbf{Y})$ $\forall \mathbf{X}, \mathbf{Y}, \mathbf{Z} \in S^+(n)$. This property is crucial for two reasons: (a) it leads to a left-covariant mean: $\mu(\mathbf{Z}\mathcal{X}) = \mathbf{Z}\mu(\mathcal{X})$[4] *i.e.*, if all poses $\mathbf{X}_i$ are transformed by $\mathbf{Z}$, the mean is transformed by $\mathbf{Z}$ as well, and (b) it ensures that the weights w_i computed in mean shift are invariant to any post-transformation $\mathbf{Z}$, leading to left-covariant mean shift.

A symmetric distance, *s.t.* $d(\mathbf{X}, \mathbf{Y}) = d(\mathbf{Y}, \mathbf{X})$ $\forall \mathbf{X}, \mathbf{Y} \in S^+(n)$, intuitively seems desirable, but its absence does not prevent a distance from being used in mean shift and furthermore, given the properties listed, it is not necessary. In other words, we do not require the distance function to be a metric. Right-invariance might also be considered a desirable property, but in the context of 3D recognition this occurrence does not relate to any meaningful behaviour.

3 The SRT Distance and Its Mean

In this section, we describe our new distance on $S^+(n)$, which fulfills all the desirable properties defined in Sect. 2.4.3. We call it the SRT distance, with corresponding mean μ_{SRT}.

3.1 Distance Definition

We first define the following component-wise distances:

$$d_s(\mathbf{X}, \mathbf{Y}) = \left| \log \left(\frac{s(\mathbf{X})}{s(\mathbf{Y})} \right) \right|, \tag{8}$$

[4] $\mathbf{Z}\mathcal{X} = \{\mathbf{ZX} : \mathbf{X} \in \mathcal{X}\}$ is a left coset of $\mathcal{X}$. Proof in [32, Append. A.4].

Table 2 Properties of distances and associated means in $S^+(n)$. †The approximation of [40] is, however, unique and translation compatible.

Properties	Extrinsic	Intrinsic	SRT
Distance:			
Symmetric	✓	✓	✗
Left-invariant	✗	✓	✓
Mean:			
Unique	✓	✗†	✓
Closed-form	✓	✗	✓
Scale-compatible	✓	✓	✓
Rotation-compatible	✗	✓	✓
Translation-compatible	✓	✗†	✓

$$d_r(\mathbf{X},\mathbf{Y}) = \|\mathbf{R}(\mathbf{X}) - \mathbf{R}(\mathbf{Y})\|_{\mathrm{F}}, \tag{9}$$

$$d_t(\mathbf{X},\mathbf{Y}) = \frac{\|\mathbf{t}(\mathbf{X}) - \mathbf{t}(\mathbf{Y})\|}{s(\mathbf{Y})}, \tag{10}$$

in which $d_s()$, $d_r()$ and $d_t()$ measure scale, rotation and translation distances respectively, with $\mathbf{X}$ and $\mathbf{Y}$ in $S^+(n)$. Given some bandwidth coefficients $\sigma_s, \sigma_r, \sigma_t > 0$, the SRT distance is defined as:

$$d_{\mathrm{SRT}}(\mathbf{X},\mathbf{Y}) = \sqrt{\frac{d_s^2(\mathbf{X},\mathbf{Y})}{\sigma_s^2} + \frac{d_r^2(\mathbf{X},\mathbf{Y})}{\sigma_r^2} + \frac{d_t^2(\mathbf{X},\mathbf{Y})}{\sigma_t^2}}. \tag{11}$$

By controlling $\sigma_s, \sigma_r, \sigma_t$, it is possible to create an SRT distance that is more sensitive to one type of transformations among scale, rotation, and translation than the others. In this sense, the SRT distance is more flexible than the Euclidean and Riemannian distances.

We now prove that the SRT distance possesses the most crucial property, the 6th property in the list.

Theorem 1. *$d_{SRT}()$ is left-invariant.*

Proof. The main idea involves showing that $d_{\mathrm{SRT}}()$ is related to a pseudo-seminorm on $S^+(n)$, *i.e.*, $d_{\mathrm{SRT}}(\mathbf{X},\mathbf{Y}) = \left\|\mathbf{Y}^{-1}\mathbf{X}\right\|_{\mathrm{SRT}}$, where

$$\|\cdot\|_{\mathrm{SRT}} = \sqrt{\frac{\log^2(s(\cdot))}{\sigma_s^2} + \frac{\|\mathbf{R}(\cdot) - \mathbf{I}\|_{\mathrm{F}}^2}{\sigma_r^2} + \frac{\|\mathbf{t}(\cdot)\|^2}{\sigma_t^2}}. \tag{12}$$

Indeed, for all $\mathbf{X}, \mathbf{Y} \in S^+(n)$, the transformation $\mathbf{Y}^{-1}\mathbf{X}$ consists of:

$$s\left(\mathbf{Y}^{-1}\mathbf{X}\right) = \frac{s(\mathbf{X})}{s(\mathbf{Y})}, \tag{13}$$

$$\mathbf{R}\left(\mathbf{Y}^{-1}\mathbf{X}\right) = \mathbf{R}^T(\mathbf{Y})\mathbf{R}(\mathbf{X}), \tag{14}$$

$$\mathbf{t}\left(\mathbf{Y}^{-1}\mathbf{X}\right) = \frac{\mathbf{R}^T(\mathbf{Y})(\mathbf{t}(\mathbf{X}) - \mathbf{t}(\mathbf{Y}))}{s(\mathbf{Y})}. \tag{15}$$

Applying the $\|\cdot\|_{\mathrm{SRT}}$ norm on $\mathbf{Y}^{-1}\mathbf{X}$ yields:

$$\begin{aligned}\left\|\mathbf{Y}^{-1}\mathbf{X}\right\|_{\mathrm{SRT}}^2 = \frac{1}{\sigma_s^2}\log^2\left(\frac{s(\mathbf{X})}{s(\mathbf{Y})}\right) + \frac{1}{\sigma_r^2}\left\|\mathbf{R}^T(\mathbf{Y})\mathbf{R}(\mathbf{X}) - \mathbf{I}\right\|_{\mathrm{F}}^2, \\ + \frac{1}{\sigma_t^2}\left\|\frac{\mathbf{R}^T(\mathbf{Y})(\mathbf{t}(\mathbf{X}) - \mathbf{t}(\mathbf{Y}))}{s(\mathbf{Y})}\right\|^2.\end{aligned} \tag{16}$$

Since the Frobenius norm is rotation invariant, the second and third terms of the right-hand side of Eq. (16) may be rewritten as:

$$\frac{1}{\sigma_r^2}\left\|\mathbf{R}^T(\mathbf{Y})\mathbf{R}(\mathbf{X}) - \mathbf{I}\right\|_{\mathrm{F}}^2 = \frac{1}{\sigma_r^2}\left\|\mathbf{R}(\mathbf{X}) - \mathbf{R}(\mathbf{Y})\right\|_{\mathrm{F}}^2, \tag{17}$$

$$\frac{1}{\sigma_t^2}\left\|\frac{\mathbf{R}^T(\mathbf{Y})(\mathbf{t}(\mathbf{X}) - \mathbf{t}(\mathbf{Y}))}{s(\mathbf{Y})}\right\|^2 = \frac{1}{\sigma_t^2}\left\|\frac{\mathbf{t}(\mathbf{X}) - \mathbf{t}(\mathbf{Y})}{s(\mathbf{Y})}\right\|^2. \tag{18}$$

proving $d_{\mathrm{SRT}}(\mathbf{X}, \mathbf{Y}) = \left\|\mathbf{Y}^{-1}\mathbf{X}\right\|_{\mathrm{SRT}}$. It follows that:

$$d_{\mathrm{SRT}}(\mathbf{X}, \mathbf{Y}) = \left\|\mathbf{Y}^{-1}\mathbf{X}\right\|_{\mathrm{SRT}} = \left\|\left(\mathbf{X}_i^{-1}\mathbf{Z}^{-1}\right)(\mathbf{Z}\mathbf{X})\right\|_{\mathrm{SRT}} = d_{\mathrm{SRT}}(\mathbf{Z}\mathbf{X}, \mathbf{Z}\mathbf{Y}), \tag{19}$$

proving $d_{\mathrm{SRT}}()$ is left invariant. □

Note that, unlike $d_{\mathrm{E}}()$ and $d_{\mathrm{R}}()$, $d_{\mathrm{SRT}}()$ is not symmetric; it could be made symmetric by a slight modification of the translation component, but at the expense of the translation-compatibility of the corresponding mean.

3.2 Mean Computation

Having defined $d_{\mathrm{SRT}}()$, we now derive the Fréchet mean μ_{SRT} using $d_{\mathrm{SRT}}()$, which is:

$$\mu_{\mathrm{SRT}}(\mathcal{X}) = \underset{\mathbf{X} \in S^+(n)}{\arg\min} \sum_i w_i d_{\mathrm{SRT}}^2(\mathbf{X}, \mathbf{X}_i). \tag{20}$$

and show that it is closed-form[5] and generally unique.

[5] Our close-form notion includes matrix singular-value decomposition.

Theorem 2. *The solution of Eq. (20), the SRT mean, is given as:*

$$s(\mu_{SRT}(\mathcal{X})) = \exp\left(\frac{\sum_i w_i \log s(\mathbf{X}_i)}{\sum_i w_i}\right), \tag{21}$$

$$\mathbf{R}(\mu_{SRT}(\mathcal{X})) = \mathrm{sop}\left(\frac{\sum_i w_i \mathbf{R}(\mathbf{X}_i)}{\sum_i w_i}\right), \tag{22}$$

$$\mathbf{t}(\mu_{SRT}(\mathcal{X})) = \sum_i \frac{w_i \mathbf{t}(\mathbf{X}_i)}{s^2(\mathbf{X}_i)} \Bigg/ \sum_i \frac{w_i}{s^2(\mathbf{X}_i)} \tag{23}$$

where $\mathrm{sop}(\mathbf{X}) = \arg\min_{\mathbf{Y}\in SO(n,\mathbb{R})} \|\mathbf{Y}-\mathbf{X}\|_{\mathrm{F}}$ *is the orthogonal projection of matrix* $\mathbf{X}$ *onto* $SO(n,\mathbb{R})$. *Additionally if* $\mathbf{X}$ *is singular-value decomposed into* $\mathbf{X} = \mathbf{U}diag(\lambda_1 \ldots, \lambda_n)\mathbf{V}^{\mathsf{T}}$ *for some orthogonal matrices* $\mathbf{U}, \mathbf{V} \in O(n,\mathbb{R})$ *and singular values* $\lambda_1 \geq \ldots \geq \lambda_n \geq 0$. *the function* $\mathrm{sop}(\mathbf{X})$ *computes*

$$\mathrm{sop}(\mathbf{X}) = \mathbf{U}diag(1, .., 1, \det(\mathbf{UV}))\,\mathbf{V}^{\mathsf{T}}. \tag{24}$$

The SRT mean is unique if and only if all the singular values are distinct.

Proof. The sum in Eq. (20) can be rewritten as

$$\sum_i w_i d^2_{\mathrm{SRT}}(\mathbf{X}, \mathbf{X}_i) = \frac{F_s(\mathbf{X})}{\sigma_s^2} + \frac{F_r(\mathbf{X})}{\sigma_r^2} + \frac{F_t(\mathbf{X})}{\sigma_t^2}, \tag{25}$$

where[6] $F_\star(\mathbf{X}) = \sum_{i=1}^N w_i d^2_\star(\mathbf{X}, \mathbf{X}_i)$. Since $s(\mathbf{X})$ only appears in $F_s(\mathbf{X})$, we can reformulate

$$s(\mu_{\mathrm{SRT}}(\mathcal{X})) = \arg\min_{s(\mathbf{X})\in\mathbb{R}^+} \sum_i w_i \log^2\left(\frac{s(\mathbf{X})}{s(\mathbf{X}_i)}\right), \tag{26}$$

yielding the solution (21). Similarly, since $\mathbf{t}(\mathbf{X})$ only appears in $F_r(\mathbf{X})$, after rewriting

$$\mathbf{t}(\mu_{\mathrm{SRT}}(\mathcal{X})) = \arg\min_{\mathbf{t}(\mathbf{X})\in\mathbb{R}^n} \sum_i w_i \frac{\|\mathbf{t}(\mathbf{X}) - \mathbf{t}(\mathbf{X}_i)\|^2}{s^2(\mathbf{X}_i)}, \tag{27}$$

we get Eq. (23) as the solution. Finally, since $\mathbf{R}(\mathbf{X})$ only appears in $F_r(\mathbf{X})$, we rewrite

$$\mathbf{R}(\mu_{\mathrm{SRT}}(\mathcal{X})) = \arg\min_{\mathbf{R}(\mathbf{X})\in SO(n,\mathbb{R})} \sum_i w_i \|\mathbf{R}(\mathbf{X}) - \mathbf{R}(\mathbf{X}_i)\|^2_{\mathrm{F}}. \tag{28}$$

This is precisely Moakher's definition of Euclidean (extrinsic) mean of 3D rotation matrices [25, def. 5.1] generalized to n-dimensional rotation matrices. Moakher proves that for the case of $n = 3$ [25, Sect. 3.1],

$$\mathbf{R}(\mu_{\mathrm{SRT}}(\mathcal{X})) = \mathrm{sop}(\overline{\mathbf{R}}) = \arg\min_{\mathbf{Y}\in SO(n,\mathbb{R})} \left\|\mathbf{Y} - \overline{\mathbf{R}}\right\|_{\mathrm{F}}, \tag{29}$$

[6] $\star$ should be replaced with s, r or t.

where $\overline{\mathbf{R}} = \sum_i w_i \mathbf{R}(\mathbf{X}_i)$, by showing that

$$\sum_i w_i \left\|\mathbf{R}(\mathbf{X}) - \mathbf{R}(\mathbf{X}_i)\right\|_{\mathrm{F}}^2 = \left(\sum_i w_i\right) \left\|\mathbf{R}(\mathbf{X}) - \overline{\mathbf{R}}\right\|_{\mathrm{F}}^2 + \sum_i w_i \left\|\overline{\mathbf{R}} - \mathbf{R}(\mathbf{X}_i)\right\|_{\mathrm{F}}^2. \quad (30)$$

which is straightforwardly generalized to the case of $n \neq 3$.

Finding $\mathrm{sop}(\overline{\mathbf{R}})$ when $n = 3$ is studied in [12, 25]. Here, we generalize the results to $SO(n, \mathbb{R})$. First, let the singular value decomposition of $\overline{\mathbf{R}}$ be

$$\overline{\mathbf{R}} = \mathbf{U}\mathrm{diag}\left(\lambda_1 .., \lambda_n\right) \mathbf{V}^{\mathsf{T}}, \quad (31)$$

for some orthogonal matrices $\mathbf{U}, \mathbf{V} \in O(n, \mathbb{R})$ and unique (but not necessarily distinct) singular values $\lambda_1 \geq \ldots \geq \lambda_n \geq 0$. Considering a change of variable $\mathbf{R}^{'} = \mathbf{U}^{\mathsf{T}}\mathbf{R}(\mathbf{X})\mathbf{V}$, we get:

$$\mathbf{U}^{\mathsf{T}}\overline{\mathbf{R}}\mathbf{V} = \mathrm{diag}\left(\lambda_1 .., \lambda_n\right), \quad (32)$$

$$\left\|\mathbf{R}(\mathbf{X}) - \overline{\mathbf{R}}\right\|_{\mathrm{F}}^2 = \left\|\mathbf{U}^{\mathsf{T}}\left(\mathbf{R}(\mathbf{X}) - \overline{\mathbf{R}}\right)\mathbf{V}\right\|_{\mathrm{F}}^2 = \left\|\mathbf{R}^{'} - \mathrm{diag}\left(\lambda_1 \ldots, \lambda_n\right)\right\|_{\mathrm{F}}^2. \quad (33)$$

Thus, minimizing $\left\|\mathbf{R}(\mathbf{X}) - \overline{\mathbf{R}}\right\|_{\mathrm{F}}^2$ with respect to $\mathbf{R}(\mathbf{X})$ is equivalent to minimizing $f(\mathbf{R}^{'}) = \left\|\mathbf{R}^{'} - \mathrm{diag}\left(\lambda_1 .., \lambda_n\right)\right\|_{\mathrm{F}}^2$ with respect to $\mathbf{R}^{'}$. Here, $\mathbf{R}^{'} \in O(n, \mathbb{R})$ and $\det(\mathbf{R}^{'}) = \det(\mathbf{UV})$. Rewriting function $f(\mathbf{R}^{'})$:

$$f(\mathbf{R}^{'}) = \mathrm{trace}\left(\mathbf{I} - 2\mathbf{R}^{'\mathsf{T}}\mathrm{diag}\left(\lambda_1 .., \lambda_n\right) + \mathrm{diag}^2\left(\lambda_1 \ldots, \lambda_n\right)\right) \quad (34)$$

$$= \sum_{i=1}^{n} \left(1 - 2\mathbf{R}^{'}_{i,i}\lambda_i + \lambda_i^2\right), \quad (35)$$

we can see that only the diagonal elements of $\mathbf{R}^{'}$ are involved in $f(\mathbf{R}^{'})$. Therefore, the optimal $\mathbf{R}^{'}$ must be a diagonal orthogonal matrix. Among all the diagonal orthogonal matrices available in $O(n, \mathbb{R})$ (there are 2^n in total), the one that minimizes $f(\mathbf{R}^{'})$ and has $\det(\mathbf{R}^{'}) = \det(\mathbf{UV})$, considering that λ_n is the smallest singular value, is given by

$$\mathbf{R}' = \mathrm{diag}\left(1, .., 1, \det(\mathbf{UV})\right). \quad (36)$$

In other words,

$$\mathrm{sop}(\overline{\mathbf{R}}) = \mathbf{U}\mathrm{diag}\left(1, .., 1, \det(\mathbf{UV})\right)\mathbf{V}^{\mathsf{T}}. \quad (37)$$

We now analyze the uniqueness of $\mathrm{sop}(\overline{\mathbf{R}})$. First, if some singular values are not distinct, *i.e.*, $\lambda_k = \lambda_{k+1}$, then the k^{th} and $(k+1)^{\text{th}}$ columns of matrices $\mathbf{U}$ and $\mathbf{V}$ of Eq. (31) become non-unique, making $\mathrm{sop}(\overline{\mathbf{R}})$ non-unique. Second, in the case that all the singular values are distinct, if $\lambda_n > 0$ then both $\mathbf{U}$ and $\mathbf{V}$ are unique, making $\mathrm{sop}(\overline{\mathbf{R}})$ unique. If $\lambda_n = 0$, the n^{th} column of $\mathbf{U}$ and the n^{th} column of $\mathbf{V}$ can both be negated while still satisfying Eq. (31) (their directions are fixed by other columns). However, $\mathrm{sop}(\overline{\mathbf{R}})$ remains unchanged due to the simultanous negation of

both singular vectors. Therefore, $\text{sop}(\overline{\mathbf{R}})$ is unique if and only if all the singular values of $\overline{\mathbf{R}}$ are distinct. □

It can be further verified that $\mu_{\text{SRT}}(\mathcal{X})$ is:

1. *Scale-compatible*: The scale component $s(\mu_{\text{SRT}}(\mathcal{X}))$ is a **geometric mean** of $s(\mathbf{X}_i)$'s.
2. *Rotation-compatible*: The rotation component $\mathbf{R}(\mu_{\text{SRT}}(\mathcal{X}))$ is an **extrinsic mean** of $\mathbf{R}(\mathbf{X}_i)$'s.
3. *Translation-compatible*: The translation component $\mathbf{t}(\mu_{\text{SRT}}(\mathcal{X}))$ is an **arithmetic mean** of $\mathbf{t}(\mathbf{X}_i)$'s.

Table 2 summarizes the desirable properties of the SRT distance and mean, and contrasts them with those of the Euclidean and Riemannian distances.

3.3 SRT Mean Shift

We form our mean shift algorithm on $S^+(n)$ using $d_{\text{SRT}}()$ and $\mu_{\text{SRT}}(\mathcal{X})$ in steps 4 & 5 of Algorithm 1 respectively. It follows from the left-invariance of d_{SRT} that SRT mean shift is left-covariant.

The coefficients $\sigma_s, \sigma_t, \sigma_r$ act in place of the kernel bandwidth σ in Eq. (5). Also note that, while the coefficient ζ is constant in Euclidean space, it is *not* constant in a non-Euclidean space, in which case $\zeta = \zeta(\mathbf{X}_i)$ [30, 40] cannot be factored out of the kernel density estimate. Since $\zeta(\mathbf{X}_i)$ can be costly to compute (sometimes non-closed-form), existing mean shift algorithms on Lie groups [7, 40] replace $\zeta(\mathbf{X}_i)$ with a constant. However, in the case of $d_{\text{SRT}}()$, indeed any left-invariant distance, it can be shown that $\zeta(\mathbf{X}_i)$ is constant:

Lemma 1. *Using d_{SRT}, the volume densities are constant:* $\forall \mathbf{X}, \mathbf{Y} \in S^+(n) : \zeta(\mathbf{X}) = \zeta(\mathbf{Y})$.

Proof. The volume density function $\zeta(\mathbf{Y})$ with respect to the SRT distance and kernel $K(\cdot)$ at transformation $\mathbf{Y}$ is given by:

$$\zeta(\mathbf{Y}) = \int_{S^+(n)} K\left(d^2_{\text{SRT}}(\mathbf{U}, \mathbf{Y})\right) d\nu(\mathbf{U}), \tag{38}$$

where $\nu(\mathbf{U})$ is a (left-)Haar measure on $S^+(n)$ [30]. $\nu(\mathbf{U})$ has a property that $d\nu(\mathbf{U}) = d\nu(\mathbf{ZU})$ for all $\mathbf{Z} \in S^+(n)$. Let us fix $\mathbf{Z} = \mathbf{XY}^{-1}$. Since $d_{\text{SRT}}()$ is left-invariant, using the substitute $\mathbf{V} = \mathbf{ZU}$ and left-multiplying both input arguments of $d_{\text{SRT}}()$ with $\mathbf{Z}$, we obtain:

$$\begin{aligned}\zeta(\mathbf{Y}) &= \int_{S^+(n)} K\left(d^2_{\text{SRT}}(\mathbf{Z}^{-1}\mathbf{V}, \mathbf{Y})\right) d\nu(\mathbf{V}) \\ &= \int_{S^+(n)} K\left(d^2_{\text{SRT}}(\mathbf{V}, \mathbf{ZY})\right) d\nu(\mathbf{V}) = \zeta(\mathbf{X}).\end{aligned} \tag{39}$$

Therefore, $\zeta(\mathbf{X})$ is constant for all $\forall \mathbf{X} \in S^+(n)$. □

4 Experiments

4.1 Experimental Setup

Our experimental data consists of 12 shape classes, for which we have both a physical object and matching CAD model. We captured the geometry of each object using Vogiatzis and Hernández's multi-view stereo method [43] in the form of point clouds (Fig. 1(b)), 20 times from a variety of poses. Along with the class label, every shape instance has an associated ground truth pose, computed by first approximately registering the relevant CAD model to the point cloud manually, then using the Iterative Closest Point algorithm [5] to refine the registration.

4.1.1 Pose Vote Computation

Given a test point cloud and set of training point clouds (with known class and pose), the computation of input pose votes $\mathcal{X}$ is a two stage process similar to [20, 41]. In the first stage, local shape features, consisting of a descriptor and a scale, translation and rotation relative to the object, are computed on all the point clouds (Fig. 1(c)). In the second stage each test feature is matched to the m (we use 20) nearest training features, in terms of Euclidean distance between descriptors, to generate m pose votes[7]

4.2 Inference

4.2.1 Mean Shift

Mean shift finds a local mode, and its weight, in the output pose distribution for a given object class. Since there may be many such modes we start mean shift from 100 random input poses for each class. Each mode, duplicates excepted, is then added to a list of candidate poses across all classes.

In $S^+(3)$ it is possible to use the quaternion representation of rotation, $\mathbf{q}(\mathbf{X})$, which we do. For efficiency, we therefore alternatively define the rotation component of $d_{\mathrm{SRT}}()$ as

$$d_r(\mathbf{X},\mathbf{Y}) = \sqrt{1 - |\mathbf{q}(\mathbf{X})^{\mathsf{T}}\mathbf{q}(\mathbf{Y})|}, \tag{40}$$

where $|\cdot|$ is needed to account for the fact that $\mathbf{q}(\mathbf{X})$ and $-\mathbf{q}(\mathbf{X})$ represent the same rotation. This formulation confers a small computational advantage over other, non-component-wise distances in this space.

[7] Since all inference methods will use the same set of input pose votes, the method by which these are computed is not central to the evaluation of relative performance.

4.2.2 Hough Voting

We implemented a published Hough voting scheme [20] to compare with the mean shift inference approaches. This computes sums of weights of the pose votes which fall into each bin of a 4D histogram over translation and scale, effectively marginalizing over rotation. The highest bin sum for each class defines a pose mode. Note that we used our own pose votes and weights, and not those computed using the method described in [20].

4.3 Evaluation

We use cross validation on our training data for evaluation—a training set is created from 19 of the 20 shape instances in each class, and the remaining instance in each class becomes a test shape. Each test shape undergoes 5 random transformations (over translation, rotation and scale in the range 0.5–2), and this process is repeated with each training shape being the test shape, creating 100 test instances per class. We use 10 classes in our evaluation (shown in Fig. 5), so 1000 tests in all. The remaining 2 classes are used to learn the optimal kernel bandwidth, σ, for each inference method. We have made the data used in this evaluation publicly available [1].

We evaluate each inference method on two criteria: Recognition rate and registration rate.

4.3.1 Recognition Rate

As described above, each inference method generates a list of modes across pose and class for a given test instance, each with an associated weight. The output class is that of the mode of highest weight. A confusion matrix logs the output class versus ground truth class across all tests. The recognition rate is given by the trace of this matrix, *i.e.*, the number of correct classifications.

4.3.2 Registration Rate

The output pose for a given test instance is given by that of the weightiest mode whose class matches the ground truth class. We choose to consider a pose $\mathbf{X}$ to be correct if its scale is within 5%, orientation is within 15° and translation is within 10% (of the object size) of the ground truth's. Explicitly, the criteria to be met are

$$\left|\log\left(\frac{s(\mathbf{X})}{s(\mathbf{Y})}\right)\right| < 0.05, \tag{41}$$

$$\operatorname{acos}\left(\frac{\operatorname{trace}(\mathbf{R}(\mathbf{X})^{-1}\mathbf{R}(\mathbf{Y})) - 1}{2}\right) < \pi/12, \tag{42}$$

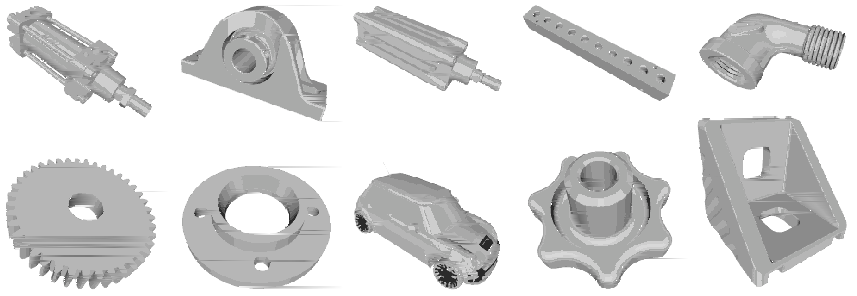


Fig. 5 Test objects. CAD models of the 10 real objects used for evaluation. *Top*: piston2, bearing, piston1, block, and pipe. *Bottom*: cog, flange, car, knob, and bracket.

$$\frac{\|\mathbf{t}(\mathbf{X}) - \mathbf{t}(\mathbf{Y})\|}{\sqrt{s(\mathbf{X})s(\mathbf{Y})}} < 0.1, \tag{43}$$

with $\mathbf{Y}$ being the ground truth pose. In the case of an object having symmetries there are multiple $\mathbf{Y}$'s, and distance to the closest is used.

4.3.3 Learning σ

We learn the mean shift kernel bandwidth, σ (or in the case of SRT, σ_s, σ_r and σ_t), used for each mean shift algorithm by maximizing the registration rate from cross-validation on two training classes (which are not used in the final evaluation). Registration rate is maximized using local search: an initial bandwidth is chosen, then the registration rate computed for this value and the values 1.2 and $1/1.2$ times this value. That value with the highest score is chosen, and the process is repeated until convergence. With 3 parameters to learn, the local search is computed over a 3D grid.

4.4 Results

Table 3 summarizes the quantitative results for the four inference methods tested. It shows that SRT mean shift performs best at both recognition and registration. The third row gives registration rate taking into account scale and translation only (as the Hough method only provides these), indicating that mean shift performs considerably better than Hough voting at registration. Also given (row 5) is the mean of output scales (each as a ratio of the output scale over the ground truth scale) of the registration result, which shows a marked bias towards a smaller scale when using extrinsic mean shift. Whilst better than extrinsic mean shift at registration, intrinsic mean shift is the slowest[8] method by an order of magnitude.

[8] We used optimized implementations for all methods.

Table 3 Quantitative results for the four inference methods tested. The SRT mean shift method is best in all respects except speed, for which it is better than the other mean shift methods.

	SRT	Extrinsic	Intrinsic	Hough
Recognition	**64.9%**	49.6%	45.5%	56.1%
Registration	**68.3%**	52.0%	62.0%	–
Registration (**t**,s)	**79.8%**	62.0%	75.7%	57.3%[9]
Processing time	1.6s	9.7s	127s	**0.043s**
Mean scale	**0.995**	0.959	0.987	–

Table 4 Registration rate per class (%). SRT mean shift performs best on 7/10 classes.

	bearing	block	bracket	car	cog	flange	knob	pipe	piston1	piston2
SRT	**77**	13	**95**	75	**100**	**41**	**88**	**86**	**44**	63
Extrinsic	36	12	90	50	80	32	53	63	37	**67**
Intrinsic	54	**19**	83	**90**	90	36	65	82	34	**67**

The per-class registration rates of the mean shift methods are given in Table 4, showing that SRT out-performs extrinsic mean shift in 9 out of 10 classes, and intrinsic mean shift in 7 out of 10. The scale-invariance of registration rate, and hence, by implication, recognition rate, using SRT and intrinsic mean shift, and the contrasting scale-variance of extrinsic mean shift (as discussed in Sect. 2.4.1), is shown empirically in Fig. 6.

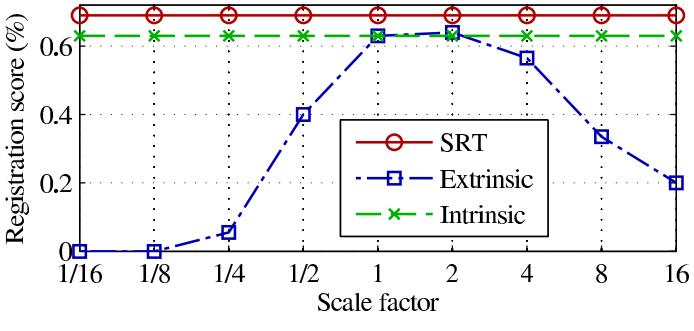


Fig. 6 Scale-invariance. Registration rate over scale, showing that only extrinsic mean shift varies with scale.

[9] This score is the percentage of ground truth poses that were in the same bin as the output pose.

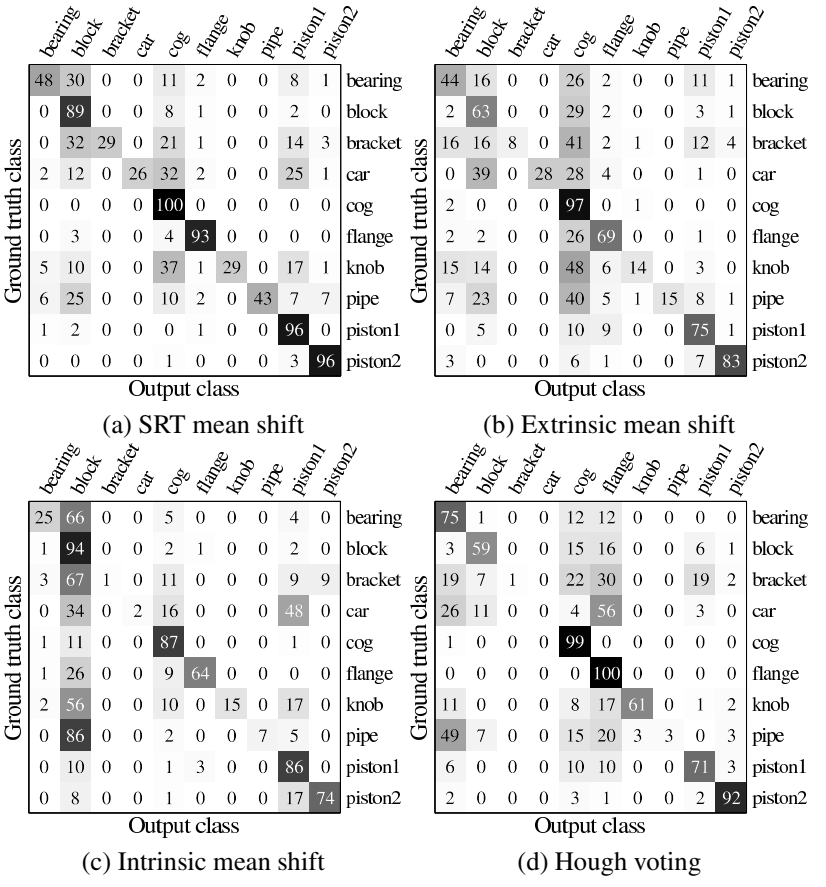
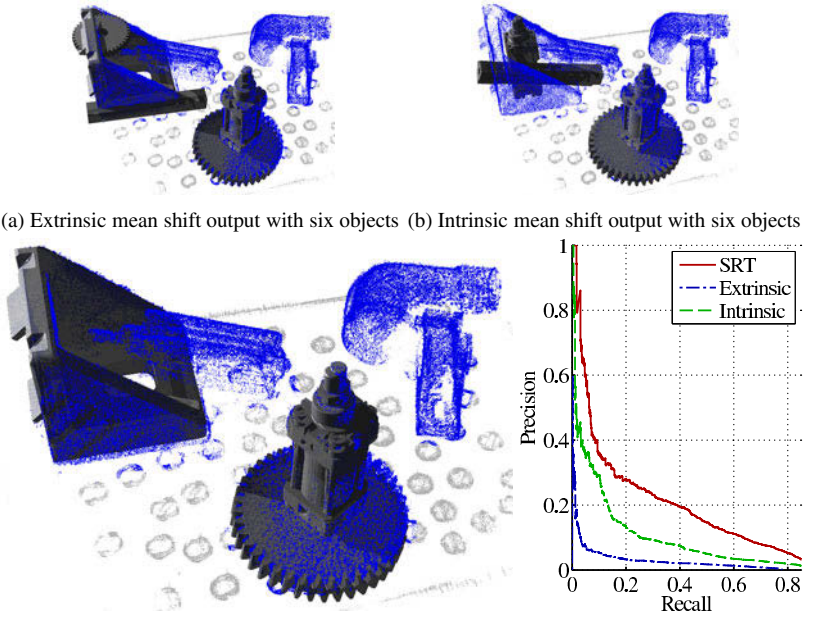


Fig. 7 Confusion matrices for the four inference methods tested. The Hough voting method performs poorly on objects with low rotational symmetry, while mean shift methods, and in particular SRT, perform better.

The confusion matrices for the four inference methods are shown in Fig. 7. Hough voting performs very poorly on bracket, car and pipe, getting a recognition rate of just 1.3% on average for these classes, which all have low rotational symmetry; in particular it prefers cog and flange (which both have high rotational symmetry), no doubt due to the marginalization this method performs over rotation. Intrinsic mean shift shows a tendency to confuse block, and cog and piston1 to a lesser degree, for other classes, whilst extrinsic and SRT mean shift confuse cog, and block and piston1 to a lesser degree for other classes.

Finally, Fig. 8a–c demonstrates that SRT mean shift applied to a real scene containing multiple objects yield more accurate results than extrinsic mean shift and intrinsic mean shift. Given a threshold weight above which modes are accepted, mean shift on the votes can produce many false positive detections, as shown by the low precision at high recall rates in Fig. 8d. This issue is addressed in another

work [44]. Our system can additionally (though not used here) filter the list of output poses using physical constraints such as the position of the ground plane and collision detection, which we found removed the majority of false positive results, including those shown in Fig. 8a–c.



(a) Extrinsic mean shift output with six objects (b) Intrinsic mean shift output with six objects

(c) SRT mean shift output with six objects (d) Precision–recall

Fig. 8 Performance with multiple objects. Given a point cloud with 6 objects, (a) Extrinsic mean shift finds 3 of them with 2 false alarms, (b) Intrinsic mean shift finds 2 of them with 2 false alarms, (c) SRT mean shift find 3 of them with no false alarms. (d) Precision-recall curves of the mean shift methods for correct registration and recognition jointly.

5 Conclusion

We have introduced the SRT distance for use in mean shift on poses in the space of direct similarity transformations, $S^+(n)$. We have proven the distance to be left-invariant, and have a unique, closed-form mean with the desirable properties of scale, rotation and translation compatibilities. We have demonstrated the use of this distance for registration and recognition tasks on a challenging and realistic 3D dataset which combines real-world objects, with and without rotational symmetries, together with a vision-based geometry capture system and basic features.

Our results show that SRT mean shift has better recognition and registration rates than both intrinsic and extrinsic mean shift, as well as Hough voting. We also show that extrinsic mean shift not only is scale-variant but also biases output scale, and

that intrinsic mean shift is slower to compute. In addition to the performance increase over Hough voting, especially in the presence of rotationally symmetric objects, we demonstrate for the first time that mean shift on the full 7D pose space of $S^+(3)$ is not only possible, but that it also provides accurate 7D registration, including rotation. This is not practical using Hough-based approaches, due to their exponential memory requirements.

Potential future research includes creating efficient probability density functions on $S^+(n)$, which will serve as building blocks for statistical learning and inference on this non-Euclidean space.

References

1. Toshiba CAD model point clouds dataset
2. Agrawal, M.: A Lie algebraic approach for consistent pose registration for general euclidean motion. In: Proc. Int. Conf. on Intelligent Robot and Systems, pp. 1891–1897 (2006)
3. Arsigny, V., Commowick, O., Pennec, X., Ayache, N.: A Log-Euclidean Polyaffine Framework for Locally Rigid or Affine Registration. In: Pluim, J.P.W., Likar, B., Gerritsen, F.A. (eds.) WBIR 2006. LNCS, vol. 4057, pp. 120–127. Springer, Heidelberg (2006)
4. Ballard, D.H.: Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognition 13(2), 111–122 (1981)
5. Besl, P., McKay, N.: A method for registration of 3D shapes. IEEE Trans. on Pattern Analysis and Machine Intelligence 14(2) (1992)
6. Campbell, R.J., Flynn, P.J.: A survey of free-form object representation and recognition techniques. Computer Vision and Image Understanding 81, 166–210 (2001)
7. Cetingul, H.E., Vidal, R.: Intrinsic mean shift for clustering on Stiefel and Grassmann manifolds. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 1896–1902 (2009)
8. Chen, H., Bhanu, B.: 3d free-form object recognition in range images using local surface patches. J. Pattern Recognition Letters 28, 1252–1262 (2007)
9. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Trans. on Pattern Analysis and Machine Intelligence 17, 790–799 (1995)
10. Davies, P.I., Higham, N.J.: A Schur-Parlett algorithm for computing matrix functions. SIAM J. Matrix Anal. Appl. 25, 464–485 (2003)
11. Drost, B., Ulrich, M., Navab, N., Ilic, S.: Model globally, match locally: Efficient and robust 3D object recognition. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 998–1005 (2010)
12. Eggert, D.W., Lorusso, A., Fisher, R.B.: Estimating 3-d rigid body transformations: a comparison of four major algorithms. Machine Vision Application 9, 272–290 (1997)
13. Ashbrook, A.P., Fisher, R.B., Robertson, C., Werghi, N.: Finding Surface Correspondence for Object Recognition and Registration Using Pairwise Geometric Histograms. In: Burkhardt, H., Neumann, B. (eds.) ECCV 1998. LNCS, vol. 1407, p. 674. Springer, Heidelberg (1998)
14. Fréchet, M.: Les éléments aléatoires de nature quelconque dans un espace distancié. Ann. Inst. H. Poincaré 10, 215–310 (1948)
15. Frome, A., Huber, D., Kolluri, R., Bülow, 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)
16. Gall, J., Lempitsky, V.: Class-specific hough forests for object detection. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 1022–1029 (June 2009)

17. Johnson, A.E., Hebert, M.: Using spin images for efficient object recognition in cluttered 3d scenes. IEEE Trans. on Pattern Analysis and Machine Intelligence 21(5), 433–449 (1999)
18. Kazhdan, M., Funkhouser, T., Rusinkiewicz, S.: Rotation invariant spherical harmonic representation of 3d shape descriptors. In: Proc. Eurographics/ACM SIGGRAPH Symp. on Geometry Processing, pp. 156–164 (2003)
19. Khoshelham, K.: Extending generalized Hough transform to detect 3D objects in laser range data. In: Workshop on Laser Scanning, vol. XXXVI, pp. 206–210 (2007)
20. Knopp, J., Prasad, M., Willems, G., Timofte, R., Van Gool, L.: Hough Transform and 3D SURF for Robust Three Dimensional Classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6316, pp. 589–602. Springer, Heidelberg (2010)
21. Leibe, B., Leonardis, A., Schiele, B.: Robust object detection with interleaved categorization and segmentation. Int. J. Computer Vision 77(1-3), 259–289 (2008)
22. Mamic, G., Bennamoun, M.: Representation and recognition of 3d free-form objects. Digital Signal Processing 12(1), 47–76 (2002)
23. Mian, A.S., Bennamoun, M., Owens, R.A.: Automatic correspondence for 3D modeling: an extensive review. Int. J. Shape Modeling 11(2), 253–291 (2005)
24. Mian, A.S., Bennamoun, M., Owens, R.: Three-dimensional model-based object recognition and segmentation in cluttered scenes. IEEE Trans. on Pattern Analysis and Machine Intelligence 28(10), 1584–1601 (2006)
25. Moakher, M.: Means and averaging in the group of rotations. SIAM J. Matrix Anal. Appl. 24, 1–16 (2002)
26. Mundy, J.L.: Object Recognition in the Geometric Era: A Retrospective. In: Ponce, J., Hebert, M., Schmid, C., Zisserman, A. (eds.) Toward Category-Level Object Recognition. LNCS, vol. 4170, pp. 3–28. Springer, Heidelberg (2006)
27. Okada, R.: Discriminative generalized hough transform for object dectection. In: Proc. Int. Conf. on Computer Vision, pp. 2000–2005 (October 2009)
28. Opelt, A., Pinz, A., Zisserman, A.: Learning an alphabet of shape and appearance for multi-class object detection. Int. J. Computer Vision 80(1) (2008)
29. Osada, R., Funkhouser, T., Chazelle, B., Dobkin, D.: Shape distributions. ACM Trans. Graph. 21, 807–832 (2002)
30. Pelletier, B.: Kernel density estimation on Riemannian manifolds. Statistics Probability Letters 73(3), 297–304 (2005)
31. Pennec, X.: Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements. JMIV 25(1), 127–154 (2006)
32. Pennec, X., Ayache, N.: Uniform distribution, distance and expectation problems for geometric features processing. J. Math. Imaging Vis. 9, 49–67 (1998)
33. Petrelli, A., Di Stefano, L.: On the repreatability of the local reference frame for partial shape matching. In: Proc. Int. Conf. on Computer Vision (2011)
34. Rusu, R.B., Blodow, N., Beetz, M.: Fast point feature histograms (fpfh) for 3d registration. In: Proc. Int. Conf. Robotics and Automation, pp. 3212–3217 (2009)
35. Saupe, D., Vranic, D.V.: 3D Model Retrieval with Spherical Harmonics and Moments. In: Radig, B., Florczyk, S. (eds.) DAGM 2001. LNCS, vol. 2191, p. 392. Springer, Heidelberg (2001)
36. Schramm, É., Schreck, P.: Solving geometric constraints invariant modulo the similarity group. In: Int. Conf. on Computational Science and Applications, pp. 356–365 (2003)
37. Shotton, J.D.J., Blake, A., Cipolla, R.: Multiscale categorical object recognition using contour fragments. IEEE Trans. on Pattern Analysis and Machine Intelligence 30(7), 1270–1281 (2008)
38. Srivastava, A., Klassen, E.: Monte Carlo extrinsic estimators of manifold-valued parameters. IEEE Trans. on Signal Processing 50(2), 299–308 (2002)
39. Subbarao, R., Meer, P.: Nonlinear mean shift for clustering over analytic manifolds. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. I, pp. 1168–1175 (2006)

40. Subbarao, R., Meer, P.: Nonlinear mean shift over Riemannian manifolds. Int. J. Computer Vision 84(1) (2009)
41. Tombari, F., Di Stefano, L.: Object recognition in 3D scenes with occlusions and clutter by Hough voting. In: Proc. Pacifc-Rim Symp. on Image and Video Technology, pp. 349–355 (2010)
42. Tombari, F., Salti, S., Di Stefano, L.: Unique signatures of histograms for local surface description. In: Proc. European Conf. on Computer Vision (2010)
43. Vogiatzis, G., Hernández, C.: Video-based, real-time multi view stereo. Image and Vision Computing 29(7), 434–441 (2011)
44. Woodford, O.J., Pham, M.-T., Maki, A., Perbet, F., Stenger, B.: Demisting the Hough transform for 3D shape recognition and registration. In: British Machine Vision Conference (2011)
45. Roger, P.: Woods. Characterizing volume and surface deformations in an atlas framework: theory, applications, and implementation. NeuroImage, 18(3):769–788 (2003)

Multiple Classifier Boosting and Tree-Structured Classifiers

Tae-Kyun Kim and Roberto Cipolla

Abstract. Visual recognition problems often involve classification of myriads of pixels, across scales, to locate objects of interest in an image or to segment images according to object classes. The requirement for high speed and accuracy makes the problems very challenging and has motivated studies on efficient classification algorithms. A novel multi-classifier boosting algorithm is proposed to tackle the multimodal problems by simultaneously clustering samples and boosting classifiers in Section 2. The method is extended into an online version for object tracking in Section 3. Section 4 presents a tree-structured classifier, called Super tree, to further speed up the classification time of a standard boosting classifier. The proposed methods are demonstrated for object detection, tracking and segmentation tasks.

1 Introduction

Boosting has become a standard method in object detection [3], tracking [26] and segmentation [33] problems, where a vast number of image sub-windows, across pixels and scales, need to be classified. Performing the tasks in a reasonable time demands extremely fast evaluation of each sub-window. A boosting classifier makes a decision by aggregating simple weak-learners such as Haar-like features, whose computations are accelerated by an integral image.

When object images exhibit multi-modalities (see Figure 1), a single boosting classifier is often not sufficient. A standard boosting classifier [25, 9] is represented by the weighted sum of binary weak-learners as

Tae-Kyun Kim
Department of Electrical and Electronic Engineering, Imperial College, London, UK
e-mail: tk.kim@imperial.ac.uk

Roberto Cipolla
Department of Engineering, University of Cambridge, UK
e-mail: cipolla@eng.cam.ac.uk

R. Cipolla et al. (Eds.): Machine Learning for Computer Vision, SCI 411, pp. 163–196.
springerlink.com

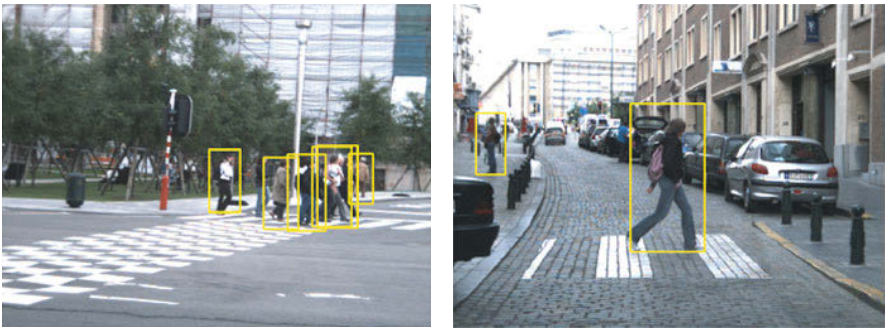


Fig. 1 Pedestrian detection. Wojek *et al.* [19] have shown that the proposed multiple classifier boosting algorithm [31] outperforms various standard methods (SVM, AdaBoost etc) for the multi-appearance pedestrian detection problems.

$$\mathrm{H}(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \mathrm{h}_i(\mathbf{x}), \tag{1}$$

where α_i is the weight and h_i the i-th binary weak-learner in $\{-1, 1\}$. Object images of e.g., multi-poses or multi-object categories are difficult to be dichotomised from non-object images by a single aggregation of simple features. Conventionally, multiple boosting classifiers, each of which is for a defined sub-category, are required [4, 6]. However, manual labeling of object categories and/or poses is difficult for a large data set and how to partition images into sub-categories is often not clear. We present a new co-clustering problem of images and visual features in Section 2. The problem is tackled by simultaneously boosting multiple classifiers which compete for object images by their expertise. Each boosting classifier is an aggregation of weak-learners, i.e., simple visual features. The solution is achieved by a gradient-descent optimisation technique. We demonstrate by both synthetic and real image data sets that the obtained classifiers are capable of solving XOR i.e., multi-modal classification problems that a standard boosting classifier fails to solve.

In object tracking a major challenge is handling appearance changes of a target object due to factors such as changing pose, illumination and deformation. Recently a class of techniques using discriminative tracking has been shown to yield good results by treating tracking as a classification framework [21, 22, 24, 26] (see Figure 2). A classifier is iteratively updated using positive and negative training samples extracted from each frame. Online boosted classifiers have been widely adopted owing to their efficiency and good classification performance [22, 26, 27]. However, as they maintain a single boosted classifier, they are limited to single view tracking or slow view changes of a target object. Tracking tends to fail during rapid appearance changes, because most weak learners of a boosted classifier do not capture the new feature distributions. Rapid adaptation of an online classifier in order to track these changes increases the risk of incorrectly adapting to background regions. Section 3 presents a new multi-pose object tracking solution by extending the multi-classifier

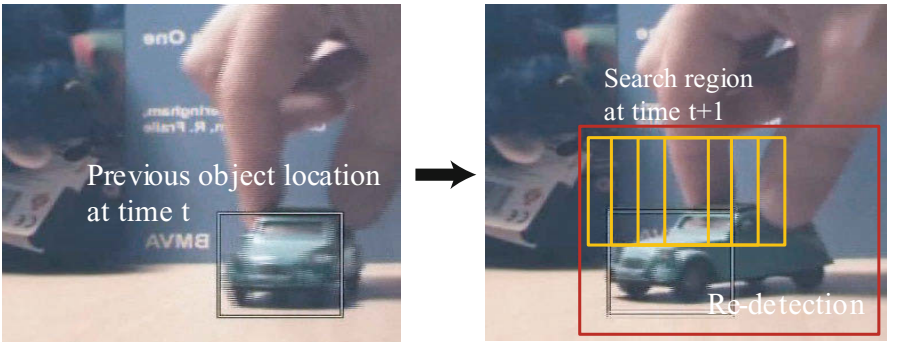


Fig. 2 Object tracking by fast re-detection. A search window is set based on the previous location and speed of a target object, and a classifier is applied to evaluate every sub-window within the search region. The proposed online multiple classifier boosting method [34] allows tracking during rapid pose changes.

boosting algorithm in Section 2 into an online version. The proposed algorithm *jointly* learns the classifiers and a soft partitioning of the input space, defining an area of expertise for each classifier. We show how this formulation improves the specificity of the strong classifiers, allowing simultaneous location and pose estimation in a tracking task. The proposed online scheme iteratively adapts the classifiers during tracking.

Despite the efficiency of a boosting classifier, it is often required to further reduce the evaluation time. A cascade of boosting classifiers, which could be seen as a degenerate tree, effectively improves the classification speed: by filtering out majority of negative class samples in its early stages [3]. Designing a cascade, however, involves manual efforts for setting a number of parameters: the number of classifier stages, the number of weak-learners and the threshold per stage. In Section 4, we present a novel way to speed up the evaluation time of a boosting classifier without needing a conventional multi-stage boosting cascade. We make a shallow (flat) network deep (hierarchical) by growing a tree from decision regions of a given boosting classifier. The obtained tree, called Super tree, provides many short paths for speeding up while it preserves the reasonably smooth decision regions of the boosting classifier for good generalisation. For converting a boosting classifier into a decision tree, we formulate a Boolean optimisation problem, which has been previously studied for circuit design but limited to a small number of binary variables. The method has been demonstrated for segmentation problems (see Figure 3).

The rest of the chapter is organised as follows: Section 2 presents the multiple classifier boosting algorithm to learn multi-modal appearances, Section 3 its online version for object tracking. Conversion of a boosting classifier into a decision tree for speeding up is explained in Section 4. The summary and conclusion is drawn in Section 5.

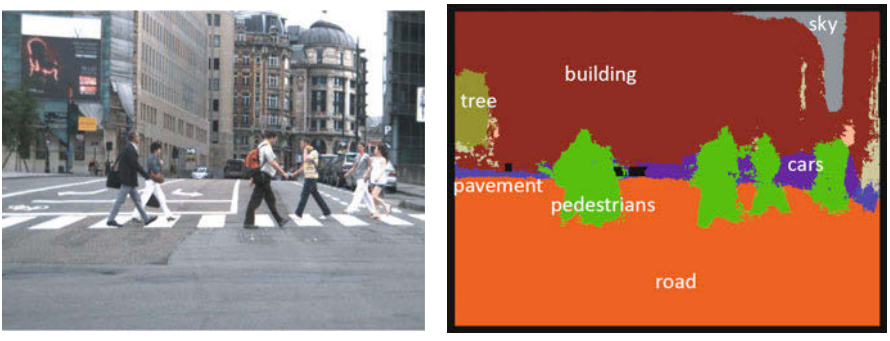


Fig. 3 Semantic segmentation. Every pixel of an input image (left) is assigned one of object categories (right).

2 Multiple Classifier Boosting

It is known that visual cells (*visual features*) selectively respond to *imagery patterns* in perception. Learning process may be associated with co-clusters of visual features and imagery data in a way of facilitating image data perception. We formulate this in the context of boosting classifiers with simple visual features for binary classification tasks e.g., object detection [3]. There are two sets of images: a set of object images and a set of non-object images, labeled as positive and negative class members respectively. There are also a huge number of simple image features, only a small fraction of which are selected to discriminate the positive class from the negative class by $\mathrm{H}(\mathbf{x}) = \sum_t \alpha_t \mathrm{h}_t(\mathbf{x})$ where $\mathbf{x}$ is an input vector, α_t, h_t are the weight and the score of t-th weak-learner using a single feature, and H is a boosting classifier. When object images exhibit multi-modalities, a single aggregation of simple features is often not sufficient to dichotomise object images from non-object images. Our problem is to find out subsets of object images, each of which is classified by an associated set of features i.e., a boosting classifier, for maximising classification performance. Note that image clusters to be obtained are coupled with selected features and likewise features to be selected are dependent on image clusters, requiring concurrent clustering of images and features.

See Figure 4 for an example where subsets of face images are pose-wise obtained with associated features by the proposed method (Section 2.1). Features are placed around eyes, a nose and mouth as the cues for discriminating faces from background. As such facial features are distributed differently mainly according to face pose in the example, the obtained pose-wise face clusters are, therefore, intuitive and desirable in perception. Note the challenges in achieving this: the input set of face images are mixed up by different persons, lighting conditions as well as poses. Some are photographs of real-faces and the others are drawings. Desired image clusters are *not observable* in input space. See Figure 5 for the clusters obtained by the traditional unsupervised clustering method (k-means clustering) on the face

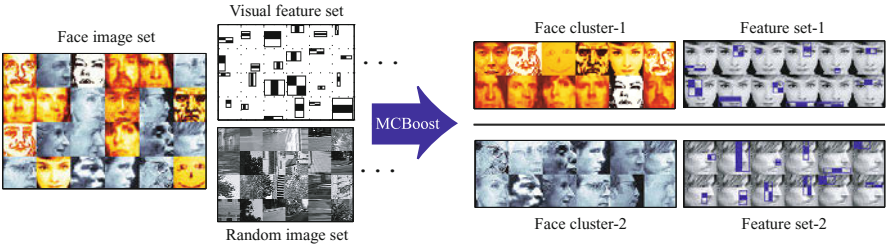


Fig. 4 Perceptual co-clusters of images and visual features. For given a set of face and random images and simple visual features, the proposed method finds the joint-clusters of face images and features, which facilitates classification of face images from random images. Face clusters are pose-wise obtained in the example.

Fig. 5 Face clusters obtained by the k-means clustering method.

images. Images of the obtained clusters are almost random with respect to pose. A required method must have a discriminative process and part-based representations (like the simple features used) to obtain more meaningful face clusters for classification. Technically, it is also required to cope with an arbitrary initialisation of image clusters because the target clusters are hidden. The k-means clustering result is completely different from that obtained by the proposed method as shown in Figure 4 and Figure 5. Feature selection should be efficiently performed among a huge number of input features.

We simultaneously boosts multiple boosting classifiers, each of which has expertise on a particular set of object images by a set of weak-learners. The proposed method (Section 2.1) has a potential for wide-applications in perceptual data exploration. It generally solves a new co-clustering problem of a data set (e.g., a set of face images) and a feature set (e.g., simple visual features) in a way to maximise discrimination of the data set from another data set (e.g., a set of random images).

Related Work

Existing co-clustering work (e.g., [1]) is formulated as an unsupervised learning task. It simultaneously clusters rows and columns of a co-occurrence table by e.g., maximising mutual information between the cluster variables. Conversely, we make use of class labels for discriminative learning. Using a co-occurrence table in prior work is also prohibitive due to a huge number of visual features that we consider.

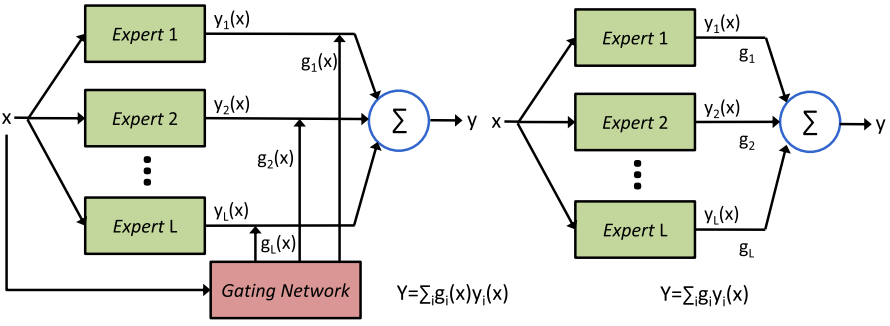


Fig. 6 Mixture of Experts vs Ensemble Learning. In MOE, the gating network as a function of input activates an expert encouraging specialization. In Ensemble learning i.e., Boosting, all experts contribute to form a decision with pre-determined weights.

Mixture of Experts [2] (MoE) jointly learns multiple classifiers and data partitions. It emphasises local experts and is suitable when input data are naturally divided into homogeneous subsets, which is, however, often not the case in practice as observed in Figure 5. Note that EM in MoE resorts to a local optimum and in practice, it is difficult to establish a good initial data partition. Furthermore, the data partitions of MoE could be undesirably affected by a large background class considered in our problem and the linear transformations used in MoE are limited for delivering a meaningful part-based representation of object images.

Boosting [8] is a sequential learning method for aggregating multiple weak classifiers. It finds weak-learners to correctly classify erroneous samples by previous weak-learners. While MoE makes a decision by dynamically selected local experts, all experts or weak-learners in a boosting classifier contribute to a decision with weights (See Figure 6). Expert selection required in MoE is generally a difficult problem when an input space is not naturally divided into sub-regions (clusters). A boosting classifier solves various non-linear classification problems but cannot solve XOR problems where only half the data can be correctly classified by each weak-learner (see [8] for the strength of weak learnability). Two disjointed sets of weak-learners, i.e., two boosting classifiers, are required to conquer each half of data by a set of weak-learners. Torralba *et al.*'s method for multiple boosting classifiers [4] relies on manual labels for cateogry/pose, whereas we optimise image clusters and boosting classifiers simultaneously.

2.1 MCBoost: Multiple Classifier Boosting

Our formulation considers K strong classifiers, each of which is represented by a linear combination of weak-learners as

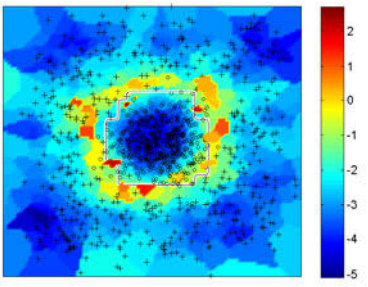


Fig. 7 Risk map computed for given two class data (circle and cross). Weak-learners (either a vertical or horizontal line) found by the Adaboost method [8] are placed on the high risk regions.

$$\mathrm{H}_k(\mathbf{x}) = \sum_t \alpha_{kt} \mathrm{h}_{kt}(\mathbf{x}), \qquad k = 1, \ldots K, \tag{2}$$

where α_{kt} and h_{kt} are the weight and the score of t-th weak-learner of k-th strong classifier. Each strong classifier is devoted to a subset of input patterns allowing repetition and each weak-learner in a classifier comprises of a single visual feature and a threshold. For aggregating multiple strong classifiers, we formulate Noisy-OR as

$$P(\mathbf{x}) = 1 - \prod_k (1 - P_k(\mathbf{x})), \tag{3}$$

where $P_k(\mathbf{x}) = 1/(1 + \exp(-\mathrm{H}_k(\mathbf{x})))$. It assigns samples to a positive class if any of classifiers does and to a negative class if every classifier does. That is, an individual classifier is forced to learn from a subset of positive samples and all negative samples. A positive sample is therefore required to be accepted as the positive class by at least one of the classifiers and a negative sample to be rejected by all. The Noisy-OR framework does not require classifier selection when making a decision: the joint probability in (3) is computed using all k classifiers for any $\mathbf{x}$. Note that the mixture of experts partitions an input space into many overlapping or disjoint regions and only classifiers of regions that a test data point falls in are used. This is a significant difference in design. A conventional design in object detection study [6] also favours the OR framework that does not need classifier selection. Our derivation builds on the previous Noisy-OR boosting algorithm [5], which has been proposed for multiple instance learning. It learns a single boosting classifier from given bags of samples whereas ours learns multiple boosting classifiers.

The sample weights are initialised e.g., by randomly partitioning positive samples, i.e., $w_{ki} = 1$ if $\mathbf{x}_i \in k$ and $w_{ki} = 0$ otherwise, where i and k denote i-th sample and k-th classifier respectively. We set $w_{ki} = 1/K$ for all k's for negative samples. For given weights, the method finds K weak-learners at t-th round of boosting, to maximise

$$\sum_i w_{ki} \cdot \mathrm{h}_{kt}(\mathbf{x}_i), \qquad \mathrm{h}_{kt} \in \mathcal{H}, \tag{4}$$

Algorithm 1. MCBoost

Input: A data set $(\mathbf{x}_i, y_i)$ and a set of pre-defined weak-learners
Output: Multiple boosting classifiers $\mathrm{H}_k(\mathbf{x}) = \sum_{t=1}^{T} \alpha_{kt} \mathrm{h}_{kt}(\mathbf{x}), k = 1..., K$

1.Compute a reduced set of weak-learners $\mathcal{H}$ by the risk map (5) and randomly initialise the weights w_{ki}.
2.Repeat for $t = 1, ..., T$:
3. Repeat for $k = 1, ..., K$:
4. Find weak-learners h_{kt} that maximise $\sum_i w_{ki} \cdot \mathrm{h}_{kt}(\mathbf{x}_i), \mathrm{h}_{kt} \in \mathcal{H}$.
5. Find the weak-learner weights α_{kt} that maximise $\mathrm{J}(\mathrm{H} + \alpha_{kt}\mathrm{h}_{kt})$.
6. Update the weights by $w_{ki} = \frac{y_i - P(\mathbf{x}_i)}{P(\mathbf{x}_i)} \cdot P_k(\mathbf{x}_i)$.
7. End
8.End

Fig. 8 Pseudocode of MCBoost algorithm

where $\mathrm{h}_{kt} \in \{-1, +1\}$ and $\mathcal{H}$ denotes a reduced set of weak-learners for speeding up the learning process. Previous methods e.g., [20] for reducing the boosting learning time may be independently deployed. The reduced set is obtained by restricting the location of weak-learners around the expected decision boundary. Each weak-learner, $h(\mathbf{x}) = \mathrm{sign}(\mathbf{a}^T\mathbf{x} + b)$, where $\mathbf{a}$ and b represent a simple feature and its threshold respectively, can be represented by $\mathbf{a}^T(\mathbf{x} - \mathbf{x}_o)$, where $\mathbf{x}_o$ is interpreted as the location of the weak-learner. By limiting $\mathbf{x}_o$ to the data points that have high risk to be misclassified, the complexity of searching weak-learners at each round of boosting is reduced. The risk is defined as

$$R(\mathbf{x}_i) = \exp\{-\frac{\sum_{j \in \mathcal{N}_i^B} \|\mathbf{x}_i - \mathbf{x}_j\|^2}{1 + \sum_{j \in \mathcal{N}_i^W} \|\mathbf{x}_i - \mathbf{x}_j\|^2}\} \tag{5}$$

where $\mathcal{N}_i^B$ and $\mathcal{N}_i^W$ are the set of predefined number of nearest neighbors of $\mathbf{x}_i$ in the opposite class and the same class of $\mathbf{x}_i$ (see Figure **??**). The weak-learner weights $\alpha_{kt}, k = 1, ..., K$ are then found to maximise $\mathrm{J}(\mathrm{H} + \alpha_{kt}\mathrm{h}_{kt})$ by a line search. Following the AnyBoost method [9, 5], we set the sample weights for the next round as the derivative of the cost function with respect to the classifier score. For the cost function $\mathrm{J} = \log \prod_i P(\mathbf{x}_i)^{y_i}(1 - P(\mathbf{x}_i))^{(1-y_i)}$, where $y_i \in \{0, 1\}$ is the label of i-th sample, the weight of k-th classifier over i-th sample is updated by

$$w_{ki} = \frac{\partial \mathrm{J}}{\partial \mathrm{H}_k(\mathbf{x}_i)} = \frac{y_i - P(\mathbf{x}_i)}{P(\mathbf{x}_i)} \cdot P_k(\mathbf{x}_i). \tag{6}$$

See Figure 8 for the pseudocode of the proposed method.

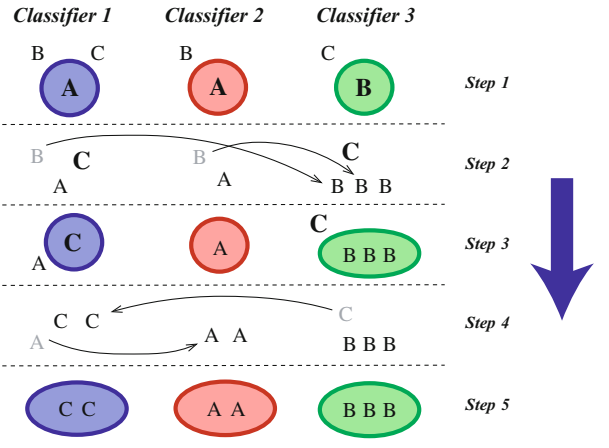


Fig. 9 State diagram for MCBoost.

Data Clustering

Data clusters (of positive samples) are obtained by assigning samples $\mathbf{x}_i$ to a classifier (or cluster) that has the highest classifier probability $P_k(\mathbf{x}_i)$.

The sample weight of k-th classifier in (6) is determined by the joint probability $P(\mathbf{x})$ and the probability of k-th classifier $P_k(\mathbf{x})$. For a negative class ($y_i = 0$), the weights only depend on the probability of k-th classifier. The classifier gives high weights to the negative samples that are misclassified by itself, independently of other classifiers. For a positive class, high weights are assigned to the samples that are misclassified jointly (i.e., the left term in (6)) but may be correctly classified by the k-th classifier through next rounds (i.e., high $P_k(\mathbf{x})$). That is, classifiers concentrate on samples in their expertise through the rounds of boosting. This can be interpreted as data partitioning.

Toy Examples

Figure 9 illustrates the concept of the MCBoost algorithm. The method iterates two main steps: learning weak-learners and updating sample weights. States in the figure represent the mode of samples (A,B or C) that are correctly classified at each step. The sample weighting (6) is represented by data re-allocation. Assume that a positive class has the samples of three target clusters denoted by A, B and C. Samples of more than two target clusters are initially assigned to a classifier. Weak-learners are found to classify the dominant mode of samples (bold letters) in each classifier (step 1). Classifiers then re-assign samples according to their expertise (step 2): Samples C that are misclassified by all are given more importance in the first classifier (bold letter). Samples B are moved to the third classifier as the expert on B. The first classifier learns next weak-learners for classifying the samples C while the second

and third classifiers focus on the samples A and B respectively (step 3). Similarly, the samples A, C are moved into the respective most experts (step 4) and all re-allocated samples are finally correctly classified by weak-learners (step 5).

We present a toy example of XOR classification problems (see Figure 10). The positive class (circle) comprising the three sub-clusters and the negative class (cross) in background make the XOR configurations. Any standard single or double boosting classifiers, therefore, do not successfully dichotomise the classes in the example. We exploit vertical or horizontal lines as weak-learners and set the number of classifiers K to be three by a priori. We performed partitioning of positive samples as shown in the left by three different color blobs (randomly mixed) for initialising the sample weights: each classifier was initially assigned all three color data points having its data center around the center of the coordinate. The final decision boundaries and the tracks of data cluster centres of the three boosting classifiers are shown in the right. Despite the mixed-up initialisation, the method learns the three classifiers that nicely settle into the target clusters after a bit of jittering in the first few rounds. The weak-learner weights (bottom) show the convergence of the three classifiers. Note that the method obtains the data clusters purely by the boosting classifier scores i.e., in a discriminative sense. Although the same data clusters are obtained by conventional clustering methods e.g., k-means in this example, clusters by conventional ways i.e., in a generative sense are often different from those of our method as exemplified in Figure 5. The proposed method works well with random initialisations and desirably exhibits quicker convergence when a better initialisation is given.

2.2 *Experiments*

Discriminative Clustering

We performed experiments using a set of INRIA pedestrian data [11] and PIE face data [10]. The INRIA set contains 618 pedestrian images as a positive class and 2436 random images as a negative class in training and 589 pedestrian and 9030 random images in testing. The pedestrian images show wide-variations in background, human pose and shapes, clothes and illuminations (Figure 11). The PIE data set involves 900 face images as a positive class (20 persons, 9 poses and 5 lighting conditions) and 2436 random images as a negative class in training and 900 face and 12180 random images in testing. The 9 poses are distributed form left profile to right profile of face, and the 5 lighting conditions make sharp changes on face appearance as shown in Figure 11. Some facial parts are not visible depending on both pose and illumination. All images were cropped and resized into 24×24 pixel images. A total number of 21780 simple rectangle features (as shown in Figure 4) were exploited.

MCBoost learning was performed with the initial weights that were obtained by the k-means clustering method. Avoiding the case that any of the k-means clusters is too small (or zero) in size has helped quick convergence in the proposed method. We set the portion of high risk data as 20% of total samples for speeding up in

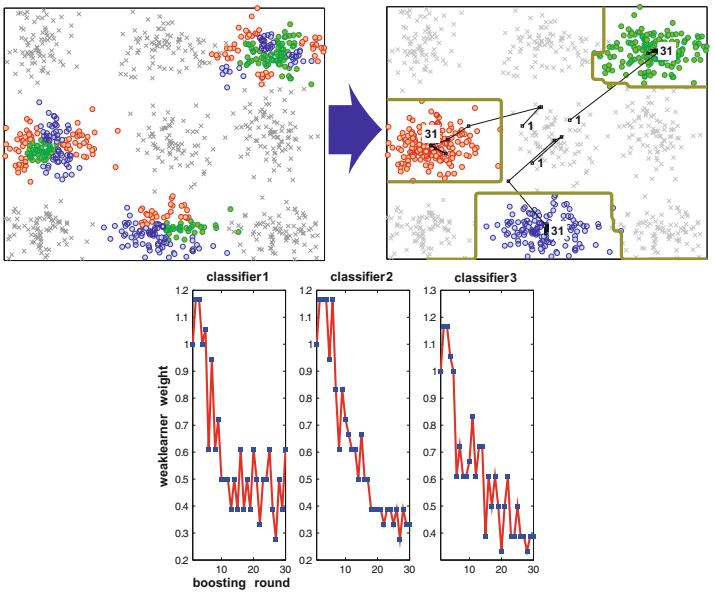


Fig. 10 Example of learning on the XOR classification problem. For a given random initialisation (three different color blobs in the left), the method learns the three classifiers that nicely settle into the desired clusters and correct decision boundaries (right). The weak-learner weights (bottom) show the convergence.

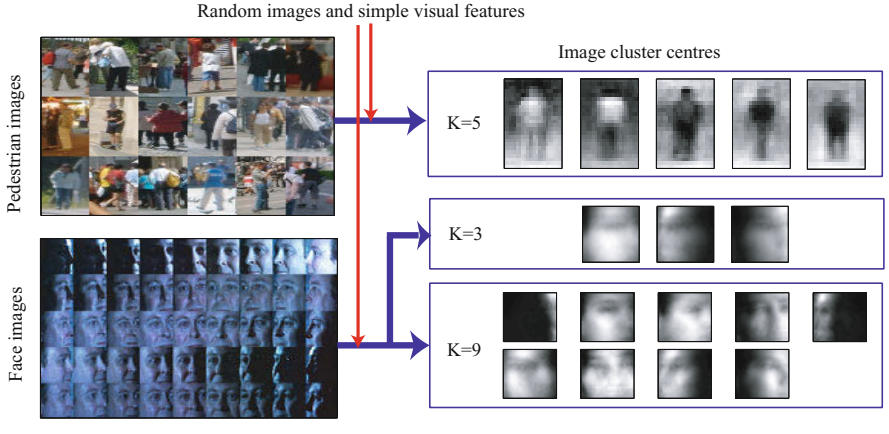


Fig. 11 Perceptual clusters of pedestrian and face images. Clusters are found to maximise discrimination power of pedestrian and face images from random images by simple visual features.

training. The number of classifiers was set as $K \in \{2,3,4,5\}$ and $K \in \{3,5,7,9\}$ for the INRIA and PIE data set respectively. For all cases, every classifier converged within 50 boosting rounds.

Figure 11 shows the cluster centers obtained by the proposed method. The object images were partitioned into K clusters (or classifiers) by assigning them to the classifier that has the highest $P_k(\mathbf{x})$. For the given pedestrian images, the first three cluster centers look unique and the last two are rather redundant. The three pedestrian clusters obtained are intuitive. They emphasise the direction of intensity changes at contours of the human body as discriminating cues of pedestrian images from random images. It is interesting to see distinction of upper and lower body in the second cluster, which may be due to different clothes. For the PIE data set, the obtained face clusters reflect both pose and illumination changes, which is somewhat different from our initial expectation of getting purely pose-wise clusters as in the case of Figure 4. This result is, however, also reasonable when considering the strong illumination conditions that shadow face parts. For example, frontal faces whose right-half side is not visible by the lighting cannot share any features with those having left-half side not visible. Certain profile faces rather share more facial features (e.g., one eye, eye brow and a half mouth) with the half-shadowed frontal faces, jointly making a cluster. All 9 face clusters seem to capture unique characteristics of the face images.

Multi-view Face Detection

Another experiment was designed using the CMU frontal and profile face image data sets [12, 13]. Some face examples were shown in Figure 4. The two data sets contain 322 images in total. The images were randomly and equally partitioned into a train and a test set. The train set of 161 images had 323 frontal faces and 192 profile faces and the test set had 271 frontal and 171 profile faces. Every face was cropped and resized into 24x24 pixel images and around 200 negative samples per image were randomly collected and resized into 24x24 pixel images. The number of negative samples in the initial train set was 32200. Two of standard AdaBoost classifiers (setting the number of weak learners be 50 per each) were initially trained using either the frontal or profile faces (by the manual pose label) with the random negative samples in the initial train set. A total number of 72000 simple rectangle features were exploited. We applied the two learnt classifiers on the train and test images for bootstrapping. The total number of bootstrapped negative samples was 7400 for the train set and 7635 for the test set. The train and test set used for comparison consisted of both frontal and profile face images and bootstrapped negative samples. The standard AdaBoost classifier (using 100 weak-learners), two AdaBoost classifiers either by the k-means clustering (K=2) or the manual pose labels (using 50 weak-learners per each) and the MCBoost classifier initialised by the k-means clustering (with K=2) (50 weak-learners per each) were compared. Figure 12 shows the ROC curves of the methods for the train (left) and test (right) sets respectively. Both graphs showed the same tendency. The MCBoost significantly outperformed the AdaBoost using 100 weak-learners (we varied the number of weak-learners and obtained the best performance by 100 weak-learners) and the AdaBoost with the k-means (K=2). The proposed method delivered the similar accuracy to that of the AdaBoost with the pose labels. The AdaBoost with the k-means outperformed the standard single

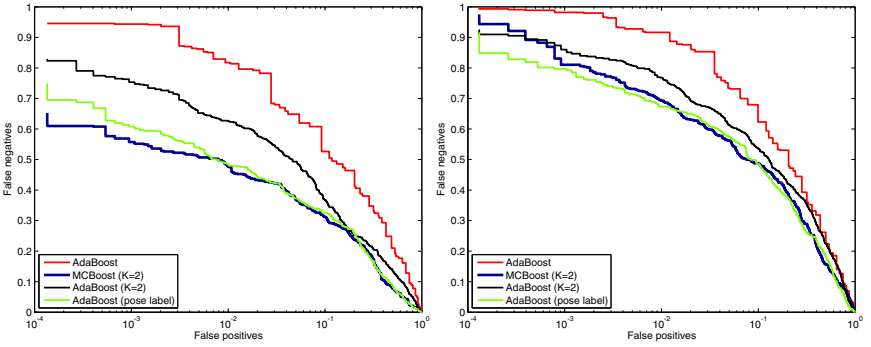


Fig. 12 ROC curves on the CMU frontal and profile face data sets. For the train set (left) and test set (right).

AdaBoost classifier. The results confirmed that the standard boosting classifier can not successfully classify samples in the XOR case (by the multi-modal face samples and bootstrapped negative samples) and the clusters learnt in the proposed discriminative manner are more suited to learn boosting classifiers than those obtained by standard unsupervised clustering methods. MCBoost exhibited very close accuracy to the classifiers learnt by the manual pose labels in the experiment.

The MCBoost method has been extensively tested by Wojek *et al.* for pedestrian detection problems in [19]. They have tested the various combinations of features (HOG, Haar, Oriented Histogram Flow) and classifiers (SVM, AdaBoost, MCBoost) on their new challenging pedestrian data sets. It has shown that MCBoost achieves superior performance to linear SVM-based detectors and significantly outperforms Adaboost for both static and dynamic pedestrian detection problems. MCBoost has been shown to be the most robust classifier with respect to challenging lighting conditions while being computationally less expensive than SVMs.

Discussions

We have introduced a discriminative co-clustering problem of images and visual features and have proposed a novel method of multiple classifier boosting called MCBoost. It simultaneously learns image clusters and boosting classifiers, each of which has expertise on an image cluster. The method works well with either random initialisation or initialisation by conventional unsupervised clustering methods. We have shown in the experiments that the proposed method yields meaningful co-clusters of images and features and significantly outperforms the conventional designs that individually learn multiple boosting classifiers by the clusters obtained by the k-means clustering method or pose-labels.

Useful future studies on the MCBoost method include development of a method to automatically determine K, the number of classifiers. At the moment, we first try a large K and decide the right number as the number of visually heterogeneous clusters obtained (see Section 2.2). A post-corrective step of initial weak-learners

would be useful for more efficient classification by less number of weak-learners in total. When the classifiers start from wrong initial clusters and oscillate between clusters until settling down, some initial weak-learners are wrong and others may be wasted to make up for the wrong ones. Once the classifiers find right clusters, they exhibit convergence by decreasing the weak-learner weights. Restaring MCBoost with the clusters found may yield economical sets of weak-learners for the same accuracy.

3 Online Multiple Classifier Boosting for Object Tracking

Object tracking has been often treated as a classification problem where it is done by fast re-detection. A search window is set based on the previous location and speed of a target object, and the object is detected within the search window. The detector is usually a binary classifier which evaluates sub-windows to tell if they contain a target object or not, at every pixel across scales, within the search window. It requires very efficient evaluation per sub-window, as it typically involves a huge number of sub-windows. Such a classification framework has been shown to yield good tracking results [21, 22, 24, 26]. A classifier is on-line updated to reflect environmental changes. Online boosted classifiers have been widely used owing to their efficiency and accuracy [22, 26, 27]. The work in [24] introduced online feature selection for tracking, where in each frame the most discriminative features are chosen to compute likelihoods. *Ensemble Tracking* [21] takes a similar approach by combining a small number of weak classifiers using AdaBoost. Online boosting for tracking [26] introduced a scheme where features are selected from a pool of weak classifiers and combined into a strong classifier. Online schemes without any target model tend to suffer from drift. One solution is to introduce an object model that is learned prior to the tracking phase [27, 29]. The work in [27] proposed semi-supervised learning, and included a boosted detector or simply the object region in the first frame as a prior to an online boosting scheme.

Maintaining a single boosted classifier during tracking is limited to single view tracking or slow view changes of a target object. Tracking tends to fail during rapid appearance changes, because most weak learners of a boosted classifier are not relevant to the new object appearance. Forcing an online classifier to adapt to these rapid changes increases the risk of incorrectly adapting to background regions. A multi-modal object representation and classifier is therefore required. Such a model can be either generative [23, 32] or discriminative [30]. Typically, in the latter case, distinct appearance clusters are found first and a classifier is trained on each [4].

Recently multi-classifier boosting was introduced, where clustering and classifier training is performed jointly [18, 31] (see Section 2.1). These methods have so far been applied to object detection, where the full training set is available from the beginning. However, direct application to the online tracking domain is not straightforward, the main reason being that in an online setting the number of positive and negative samples is not sufficient to ensure a good partitioning of the input space

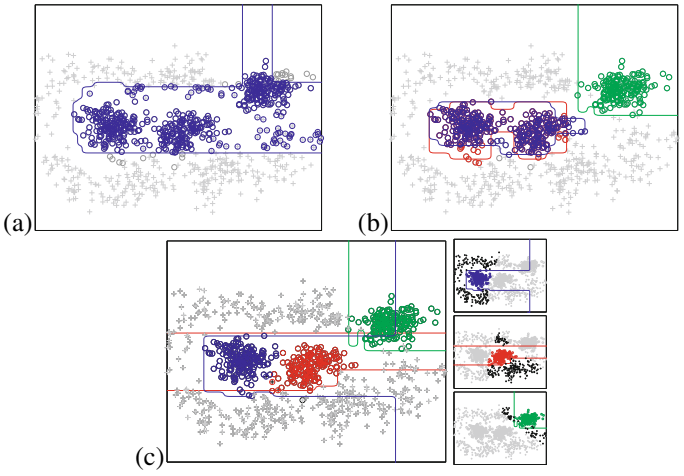


Fig. 13 Learning cluster-specific classifiers on toy data. The positive class (circles) exhibits three clusters and is surrounded by data from the negative class (crosses). (a) The classification result using a standard boosting classifier shows errors due to the XOR configuration (colored circles denote classification as positive class). (b) The Multi-classifier boosting algorithm of [31] successfully divides the two classes but uses two boosting classifiers (blue and red line) in the same region, leading to inefficient use of weak classifiers. The two clusters with no negative data points between them can be correctly classified by a single boosting classifier. (c) The classification result of the proposed MCBQ algorithm shows improved classifier expertise.

in terms of classifier expertise in the initial phase. Figure 13 illustrates the classification results of (a) standard Adaboost [25], (b) MCBoost [31] and (c) the proposed algorithm called MCBQ on a toy XOR classification problem. The positive class exhibits three clusters, but two of them actually form a single cluster in a discriminative sense as there are no negative points between them. Standard AdaBoost shows poor separation of the classes because it is unable to resolve XOR configurations. For the MCBoost algorithm and the proposed solution, we set the number of classifiers to be three. MCBoost successfully divides the two classes but shows overlapping areas of expertise for the two classifiers, since the two clusters without negative data points in-between can be correctly classified by a single boosting classifier. In contrast, the proposed algorithm shows improved partitioning of the input space. As a consequence, weak classifiers are used more efficiently. While tracking continues, additional negative samples are collected, eventually establishing three positive clusters in a discriminative sense in this example. However, in the case of MCBoost, the initially incorrectly assigned boosting classifiers are difficult to be correctly reassigned during online updates. We have observed this case when classifiers are initially trained on a short sequence that contains multi-views of a target object and are subsequently updated.

We therefore propose an extension of the multi-classifier boosting algorithm by introducing a weighting function $\mathcal{Q}$ that enforces a soft split of the input space.

In addition, we present an online version of the algorithm to dynamically update the classifiers and the partitioning for the task of multi-modal object tracking. The algorithm is applied to object tracking where it is used to learn different appearance clusters during a short initial supervised learning phase.

Other related work is online multiple instance learning (MIL) [22, 5]. Our proposed method can be seen as a multi-class extension of [22].

3.1 Joint Boosting and Clustering

This section explains our improvements in the MCBQ algorithm, based on the multi-classifier boosting algorithm in Section 2. The following notation is used: Given is a set of n training samples $\mathbf{x}_i \in \mathcal{X}$, where $\mathcal{X}$ is the input domain (in our case image patches), with labels $y_i \in \{-1,+1\}$ corresponding to non-object and object, respectively. Additionally, each of the object samples can be considered belonging to one of K groups where the class membership is a priori unknown.

Multi-classifier Boosting creates strong classifiers with different areas of expertise. However, it relies on the training data set containing negative samples which separate the positive samples into distinct regions in the classifiers' discriminative feature space. This also implies that there is no guarantee of pose-specific clustering. In fact there is no constraint in the algorithm that enforces strong classifiers to focus on a unique area of expertise, and there is no concept of a metric space on which perceived clusters can be formed. We make the classifier assignment explicit by defining functions $\mathcal{Q}^k(\mathbf{x}_i) : \mathcal{X} \rightarrow [0,1]$ which weight the influence of strong classifier k on a sample $\mathbf{x}_i$. By mapping $\mathbf{x}_i$ into a suitable metric space, we can impose any desired clustering regime on the training set, thus $\mathcal{Q}$ defines a soft partitioning of the input space. The choice of $\mathcal{Q}$ is dependent on the application domain. In principle any function can be used that captures the structure of the input domain, i.e., that maps the samples to meaningful clusters. In this method $\mathcal{Q}$ is defined by a K-component Gaussian mixture model in the space of the first d principal components of the training data. The k-th GMM mode defines the area of expertise of the k-th strong classifier. The GMM is updated using a EM-like algorithm alongside the weak classifiers in the boosting algorithm (Algorithm 1).

The new noisy-OR function in Equation 3 becomes:

$$p(\mathbf{x}_i) = 1 - \prod_k (1 - \mathcal{Q}^k(\mathbf{x}_i)\, p^k(\mathbf{x}_i)), \tag{7}$$

leading to the new weight update equation:

$$w_i^k = \frac{\partial \mathcal{L}}{\partial \mathrm{H}^k(\mathbf{x}_i)} = \frac{y_i - p(\mathbf{x}_i)}{p(\mathbf{x}_i)} \frac{\mathcal{Q}^k(\mathbf{x}_i)\, p^k(\mathbf{x}_i)\,(1 - p^k(\mathbf{x}_i))}{1 - \mathcal{Q}^k(\mathbf{x}_i)\, p^k(\mathbf{x}_i)}. \tag{8}$$

The full MCBQ algorithm is summarized in Algorithm 2. Note that compared to the original multi-classifier boosting algorithm additional steps 1, 2, and 8 are required and step 7 is modified.

Algorithm 1. Updating Weighting Function

1. Calculate the likelihood of each of the samples under the k-th strong classifier, $p^k(\mathbf{x}_i)$

2. Set the new probability of the sample being in the k-th GMM component as its current $\mathcal{Q}$ value scaled by the likelihood from the classifier, $\mathcal{Q}^k(\mathbf{x}_i)p^k(\mathbf{x}_i)$

3. Update the k-th cluster by the mean and covariance matrix of the samples under this probability.

Algorithm 2. Multi-classifier Boosting with Weighting Function (MCBQ)

Input: Data set $(\mathbf{x}_i, y_i)$, set of pre-defined weak learners.
Output: Multiple strong classifiers $H^k(\mathbf{x}_i)$, weighting function $Q^k(\mathbf{x}_i)$.
1. Initialize Q with a Gaussian mixture model.
2. Initialize weights w_i^k to the values of $Q^k(\mathbf{x}_i)$.
3. Repeat for $t = 1, ..., T$
4. Repeat for $k = 1, ..., K$
5. Find weak learners h_t^k maximizing $\sum_i w_i^k \mathrm{h}_t^k(\mathbf{x}_i)$.
6. Compute weights α_t^k maximizing $\mathcal{L}(\mathrm{H}^k + \alpha_t^k \mathrm{h}_t^k)$.
7. Update weights by Equation 8.
8. Update weighting function $Q^k(\mathbf{x}_i)$ by Algo 1.
9. End
10. End

3.2 Online MCBQ for Object Tracking

The goal is to learn an object-specific appearance model using a short initial training sequence in order to guide the tracker [27, 32]. The number of training samples is limited, but is sufficient to bootstrap the classifier. Subsequently, we would like the tracker to remain flexible to some appearance changes while using the learned model as an anchor. This motivates the following approach of iteratively adapting multiple strong classifiers with MCBQ. In order to move MCBQ into an online setting we need a mechanism for rapid feature selection and incremental updates of the weak classifiers as new training samples become available. The online boosting algorithm [26] addresses this issue, allowing for the continuous learning of a strong classifier from training data. The key step is, at each boosting round, to maintain error estimates from samples seen so far, for a pool of weak classifiers. At each round t a *selector* S_t maintains these error estimates for weak classifiers in its pool, and chooses the one with the smallest error to add to the strong classifier.

To summarize, our tracking algorithm contains two-stages: Firstly, training data is assembled in a supervised learning stage, where the system is given initial samples which span the extent of all appearances to be classified. An initial MCBQ

classifier is then built rapidly from this data. Secondly, additional training samples are supplied to update the classifier with new data during tracking.

Weak Learning and Selection

All weak classifiers use a single Haar-like feature. For online learning from a feature f and labeled samples $(\mathbf{x}_i, y_i)$ we create a decision threshold θ_m^k with parity p_m^k from the mean of feature values seen so far for positive and negative samples, where each feature value is weighted by the corresponding image weight:

$$h_{t,m}^k(\mathbf{x}_i) = p_m^k \operatorname{sign}(f(\mathbf{x}_i) - \theta_m^k), \tag{9}$$

$$\theta_m^k = (\mu^{k,+} + \mu^{k,-})/2, \quad p_m^k = \operatorname{sign}(\mu^{k,+} - \mu^{k,-}), \tag{10}$$

$$\mu^k = \frac{\Sigma_i |w_i^k| f(\mathbf{x}_i)}{\Sigma_i |w_i^k|}. \tag{11}$$

The error of the weak classifier is then given as the normalized sum of the weights of mis-classified samples:

$$e_{t,m}^k = \frac{\sum_i \mathbf{1}(h_{t,m}^k(x_i) \neq y_i)|w_i^k|}{\sum_i |w_i^k|}. \tag{12}$$

A weak classifier can then be chosen from a pool as the one giving the minimum error.

Supervised Learning

During the supervised learning stage, we have a set of weighted samples, and a global feature pool $\mathcal{F}$. Weight distributions are initialized to randomly assign positive samples to a strong classifier k, and at each round t and strong classifier k the equations 9, 10, 11, 12 are applied to initialize and select a weak classifier based on exact errors. In order to facilitate selection at the incremental update stage, we store in each selector S_t^k, for the positive and negative samples (1) for each feature value, the sum of weights of samples with that value, and (2) the sum of image weights.

To improve speed, each selector only keeps the best M performing weak classifiers for use in the incremental update stage. After each round of boosting, image weights are updated as in Equation 8, and voting weights calculated based on the error of the chosen weak classifier.

Incremental Update

Once the initial classifier has been created, it can be updated with new samples. Weights for positive samples are initialized based on their classification responses from each of the component strong classifiers in the MCBQ classifier, and the

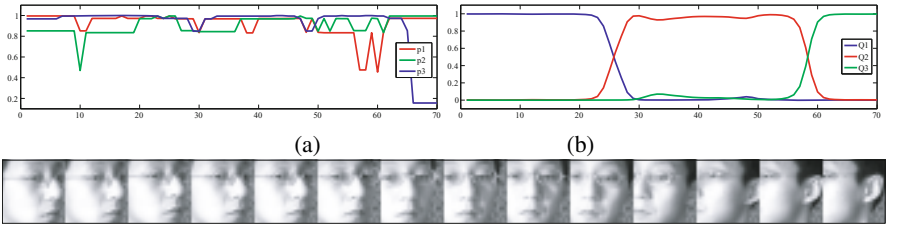


Fig. 14 Improved pose expertise: Plots of the contributions of three strong classifiers given the image input (bottom row). (a) MCBoost [31] shows no clear separation of expertise over different poses, while (b) MCBQ has learned pose-specific classifiers, corresponding to left, center and right view of the face.

sample is passed through the boosting framework. The summations stored in each selector can be updated from the new sample, and thus the new classification thresholds for the weak classifiers calculated using equations 9, 10, 11. The error values from Equation 12 are used to choose the best weak classifier to add to the strong classifier. Finally, the worst-performing weak classifier is replaced with a new randomly-generated one. Note that in the case of $\mathcal{Q}$ being defined as a Gaussian mixture in PCA space, we update the PCA space by the algorithm of Hall *et al.* [28] before updating $\mathcal{Q}$. Pseudo-code is given in Algorithm 3.

3.3 Results

Pose Clustering

For this experiment we captured short training and testing sequences (about 100 frames each) of a face rotating from left to right, see Fig. 14. We trained classifiers using MCBoost [31] and the MCBQ algorithm on face images and random patches sampled from the training sequence. In both cases the number of strong classifiers K is set to 3 by hand. The $\mathcal{Q}$ function is defined by a 3-component Gaussian mixture on the first 30 principal components. The graph in Fig. 14 shows the contribution of each strong classifier on the test sequence. The MCBoost algorithm shows no clear pose-specific response, while MCBQ has successfully captured three distinct pose clusters, left, right, and center, as shown by the changes in classifier weights.

Tracking Performance

In order to evaluate the performance on the multi-appearance tracking problem, we captured four sequences where the target object rapidly changes its pose. The sequences are *toyface* (452 frames), *handball* (210 frames), *cube* (357 frames), and *face* (185 frames). We also compared on the public *Sylvester* sequence (1345 frames). The performance was evaluated against manually labeled ground truth. We compared AdaBoost, MCBoost and MCBQ trackers (both manually set to $K = 2$),

Algorithm 3. Online MCBQ – Incremental Update

Require: Labeled training image $(\mathbf{x}_i, y_i)$, $y_i \in \{-1, +1\}$.
Require: MCBQ classifier $\mathrm{H}^k(\mathbf{x}_i)$, $k = 1, ..., K$.
// Initialize sample weight $w_i^k = \mathcal{Q}^k(\mathbf{x}_i) / \sum_k \mathcal{Q}^k(\mathbf{x}_i)$

// For each round of boosting
for $t = 1, \ldots, T$ **do**
  // For each strong classifier, update selector $\mathbf{S}_t^k$
  for $k = 1, \ldots, K$ **do**
    // Update the selector's weak classifiers
    for $m = 1, 2, \ldots, M$ **do**
      // Update cached weight sums from sample's feature value, for positive and negative samples
      // Update classification threshold and parity $\left(h_{t,m}^k, (\mathbf{x}_i, y_i), w_i^k\right)$
      // Calculate new error $e_{t,m}^k = \sum_i \mathbf{1}(h_{t,m}^k(\mathbf{x}_i) \neq y_i) |w_i^k|$
    end for
    // Choose the weak classifier with the lowest error
    $m^* = \mathrm{argmin}_m \left(e_{t,m}^k\right)$, $h_t^{k*} = h_{t,m^*}^k$ and $e_t^{k*} = e_{t,m*}^k$
    // Calculate voting weight $\alpha_t^k = 1 / \left(1 + \exp\{-\ln\left(\frac{1-e_t^{k*}}{e_t^{k*}}\right)\}\right)$
    // Replace the weak classifier with the highest error
    $m^- = \mathrm{argmax}_m \left(e_{t,m}^k\right)$ and replace h_{t,m^-}^k
  end for

  // Update $\mathcal{Q}^k(\mathbf{x}_i)$ function
  // Update importance weights by Equation 8, then re-normalize.
end for

as well as two publicly available trackers, Semi-supervised Boosting [27] and MIL tracking [22]. For each sequence the initial classifier was trained on a short initial training set (25-40 frames), capturing the appearance variation, and updated online during tracking. Examples of positive training samples are shown in Fig. 15. Because such a training set is generally not available for public tracking sequences, the training data for the *Sylvester* sequence was constructed by randomly sampling 30 frames from the whole sequence. For AdaBoost, MCBoost and MCBQ 50 random patches per frame were collected as negative class samples. We stopped boosting rounds when the classification error reached zero on the training samples. The public code for semi-supervised Boosting and MIL tracking was modified so that these methods can also be trained on the initial set, otherwise their default parameters were used. Parameter settings were unchanged for all experiments. Fig. 16 shows the tracking errors on the five sequences. While none of the algorithms was able to successfully track the target in all sequences, MCBQ showed the best overall performance, in particular outperforming AdaBoost and MCBoost. The MIL tracker performed best on two sequences, however, was not able to recover from drift in two

Fig. 15 Positive class samples for training. A subset of the positive samples is shown for the four sequences.

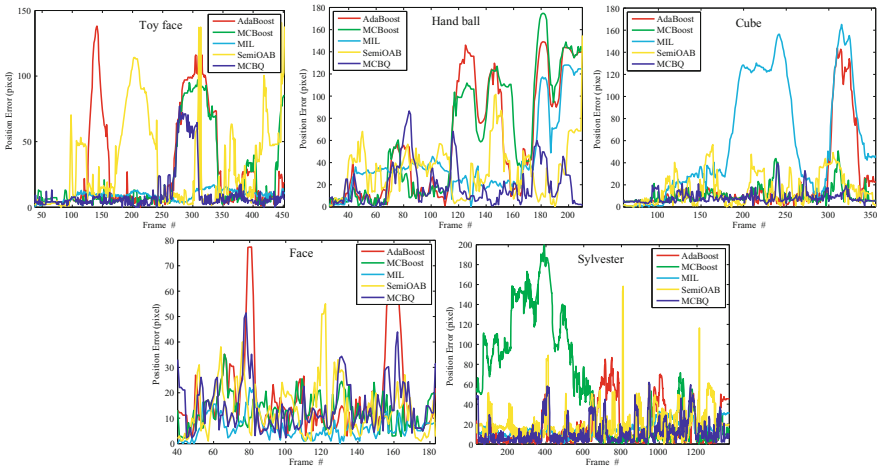


Fig. 16 Tracking error on test sequences. The plots show the tracking error over time on four test sequences for AdaBoost (red) MCBoost (green), MCBQ (blue), MILTrack (cyan), and SemiBoost (yellow). MCBQ shows the best overall performance.

of the other sequences. Overall, the single classifier trackers tend to adapt to a current appearance mode forgetting previous appearance modes, which often makes them fail when target objects rapidly change appearance modes. Fig. 17 shows example frames from the test sequences.

Discussions

This section proposed MCBQ, a multi-classifier boosting algorithm with a soft partitioning of the input space. This is achieved with a weighting function $\mathcal{Q}$ ensuring that coherent clusters are formed. We applied the method to simultaneous tracking and pose estimation. The learned model allows tracking during rapid pose changes, since it captures multiple appearances. Existing single classifier trackers tend to adapt to a single appearance mode, forgetting previous modes. MCBQ can be seen as an extension of MCBoost [31] for the online setting, or a multi-class extension of the MIL tracker [22]. Future work includes a more principled selection of the number of strong classifiers and exploring other choices for the weighting function.

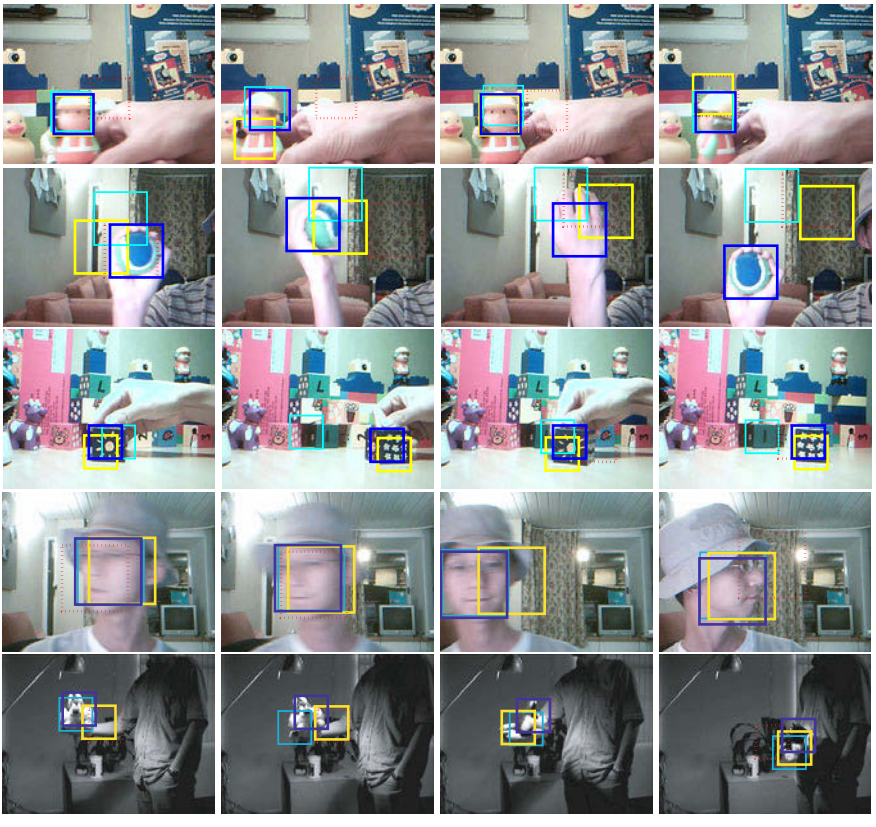


Fig. 17 Example tracking results on test sequences. The comparison shows tracking results for MCBQ (blue), AdaBoost (red), MILTrack (cyan), and SemiBoost (yellow) in the evaluation. See text for details.

4 Conversion of a Boosting Classifier into a Decision Tree by Boolean Optimisation

Boosting is a popular method in object detection [3], tracking [26] and segmentation [33] tasks, which demand very fast classification. Boosting makes a decision by aggregating simple weak-learners e.g., Haar-like features, which are computed very fast on an integral image. Despite its efficiency, it is often required to further reduce the classification time. A cascade of boosting classifiers, which could be seen as a degenerate tree (see Figure 18(a)), effectively improves the classification speed: by filtering out majority of negative class samples in its early stages [3]. Designing a cascade, however, involves manual efforts for setting a number of parameters: the number of classifier stages, the number of weak-learners and the threshold per stage.

In this work, we propose a novel way to reduce down the classification time of a boosting classifier up to an order of magnitude without sacrificing its accuracy,

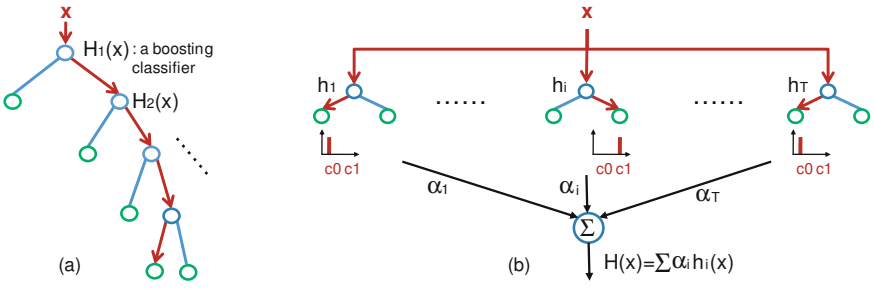


Fig. 18 Boosting as a tree. (a) A boosting cascade is seen as an imbalanced tree, where each node is a boosting classifier. (b) A boosting classifier has a very shallow and flat network where each node is a decision-stump i.e., weak-learner.

not relying on a design of cascade. The chance for improvement comes from the fact that a standard boosting classifier can be seen as a very shallow network, see Figure 18(b), where each weak-learner is a decision-stump and all weak-learners are used to make a decision. The flat structure ensures reasonably smooth decision regions for generalisation, however it is not optimal in classification time. The proposed method converts a shallow network (a boosting classifier as input) to a deep hierarchical structure (a decision tree as output). The obtained tree speeds up a boosting classifier by having many short paths: easy data points are classified by a small number of weak-learners. Since it preserves the same decision regions of the boosting classifier, the method alleviates a highly-overfit behaviour of conventional decision trees. We introduce a novel Boolean optimisation formulation and method. A boosting classifier splits a data space into 2^n primitive regions by n binary weak-learners. The decision regions of the boosting classifier are encoded by the boolean codes and class labels of the primitive regions. A decision tree is then grown using the region information gain. Further details are about a better way of packing the region information and the two stage cascade allowing the conversion with any number of weak-learners. Without designing a many-stage cascade our method offers a convenient way of speeding up, while the method incorporated in such a cascade could provide a further speed-up.

Related Work

For speeding up the classification of a boosting classifier, the shortest set of weak-learners for a given error rate has been obtained by the sequential probability ratio test in the work of Sochman *et al.* [7]. It takes an early exit when the boosting sum reaches a certain value whose sign cannot be altered by the remaining weak-learners. Similarly, Zhou has proposed Fast exit method [35]. This line of methods utilises so called *a single path of varying length*, while our tree method *multiple paths of different lengths* (See Figure 19). The proposed method yields a more optimal speed (see Section 4.2).

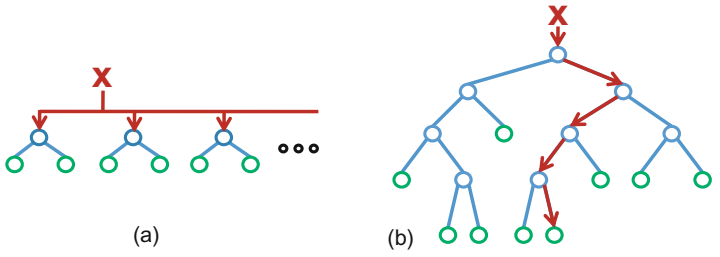


Fig. 19 Fast-exit vs super tree. Fast-exit methods have the structure of a single path of varying lengths (a), while our method yields the structure of multiple paths of different lengths (b).

The closest work to ours is Zhou's [35]. He has introduced representation of a boosting classifier by a Boolean table and implemented a binary decision tree [35]. His solution, however, is a brute force search for all possible tree configurations, which is highly computationally-costly. It therefore affords to only about 5 and 10 weak-learners. The speed gain reported was not significant over a standard boosting classifier and Fast exit method.

Tree-structured multiple boosting classifiers have been proposed for multi-pose or multi-category detection problems. The common structure is a tree hierarchy each path of which is a strong boosting classifier. Torralba *et al.* have proposed sharing weak-learners among multiple boosting classifiers [4] for accelerating classification speed. While Torralba's method requires pre-defined sub-category labels, the methods in [14, 15, 16] automatically learn the sub-category labels for multiple boosting classifiers in a tree. Whereas all these methods are useful for *multiple* boosting classifiers, our work focuses on a *single* boosting classifier. A further conceptual difference lies in that the previous studies [17, 14, 15, 16] present a novel way of learning boosting classifiers and ours takes a boosting classifier learnt in a standard way as input. We do not alter the decision regions of an input classifier but speed it up.

Boolean expression minimisation is to minimize the number of terms and binary variables in the Boolean expression. Algorithms for the minimisation have mainly been studied in the circuit design [38]. Since circuits have strictly predefined specifications, exact minimization was the goal of most studies. The complexity of a logic expression rises exponentially when the number of binary variables increases. Therefore, conventional minimisation methods are limited to a small number of binary variables, typically from a few to about 15 variables [38]. Boolean minimisation has been also applied to size down a redundant decision tree, represented by a Boolean table [39].

4.1 Conversion of a Boosting Classifier into a Tree

Both a boosting classifier and a decision tree are composed of weak-learners (or called decision-stumps/split-nodes). Whereas a boosting classifier places decision

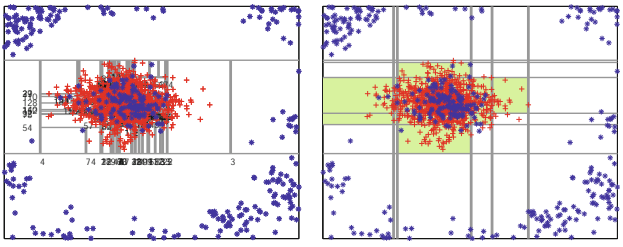


Fig. 20 Decision regions. The decision regions of a boosting classifier (right) are smooth compared to those of a conventional decision tree (left).

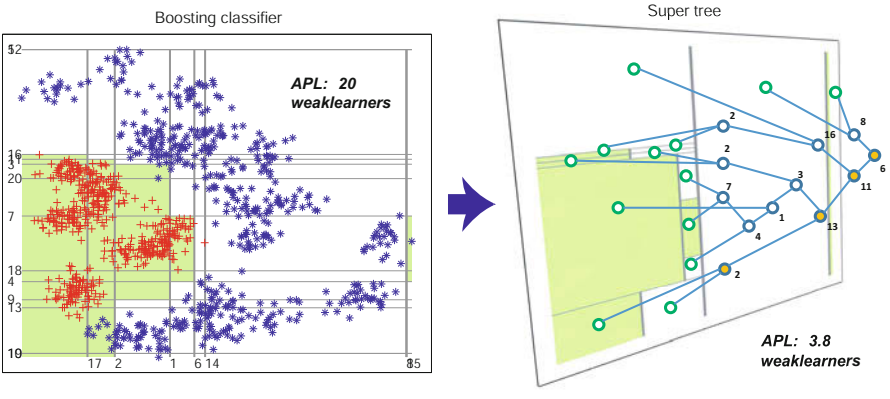


Fig. 21 Converting a boosting classifier into a tree for speeding up. The proposed conversion preserves the Boosting decision regions and has many short paths speeding up 5 times.

stumps in a flat structure, a decision tree has a deep and hierarchical structure (see Figure 18(b) and 21). The different structures lead to different behaviours: Boosting has a better generalisation via reasonably smooth decision regions. See Figure 20 for the decision regions of the two methods. Here a part of negative (blue) data points are scattered in the middle of positive (red) samples. Whereas a conventional decision tree forms complex decision regions trying classification of all training points, a boosting classifier exhibits a reasonable smoothness in decision regions. We propose a method to grow a tree from the decision regions of a boosting classifier. As shown in Figure 21, the tree obtained, called *super tree*, preserves the Boosting decision regions: it places a leaf node on every region that is important to form the identical decision boundary (i.e., accuracy). In the mean time, Super tree has many short paths that reduce the average number of weak-learners to use when classifying a data point. In the example, super tree on average needs 3.8 weak-learners to perform classification whereas the boosting classifier needs 20: all 20 weak-learners are used for every point.

4.1.1 Boolean Optimisation Formulation

A standard boosting classifier is typically represented by the weighted sum of binary weak-learners as

$$\mathrm{H}(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \mathrm{h}_i(\mathbf{x}), \tag{13}$$

where α_i is the weight and h_i the i-th binary weak-learner in $\{-1, 1\}$. The boosting classifier splits a data space into 2^m primitive regions by m binary weak-learners. Regions $R_i, i = 1, ..., 2^m$ are expressed as boolean codes (i.e., each weak-learner h_i corresponds to a binary variable w_i). See Figure 22 for an example, where the boolean table is comprised of 2^3 regions. The region class label c is determined by Equation 13. Region R_8 in the example does not occupy the 2D input space and thus receives the *don't care* label marked "x" being ignored when representing decision regions. The region prior $p(R_i)$ is introduced for data distribution as $p(R_i) = M_i/M$ where M_i and M are the number of data points in the i-th region and in total. The decision regions of the boosting classifier are encoded by a set of regions represented as

$$\begin{cases} B(R_i): \text{ boolean expression} \\ c(R_i): \text{ region class label} \\ p(R_i): \text{ region prior} \end{cases} \tag{14}$$

With the region coding, an optimally short tree is defined in terms of average expected path length of data points as

$$\mathbf{T}^* = \min_{\mathbf{T}} \sum_i E(l_{\mathbf{T}}(R_i)) p(R_i), \tag{15}$$

where $\mathbf{T}$ denotes all possible configurations of a decision tree. $E(l_{\mathbf{T}}(R_i))$ is the expected path length of the i-th region in $\mathbf{T}$. The path length is simply the number of weak-learners (or split-nodes) on the path to the i-th region. The decision tree should closely duplicate the decision regions of the boosting classifier as an optimisation constraint: the regions that do not share the same class label $c(R_i)$ must not be put in the same leaf-node of the tree. Any regions of *don't care* labels are allowed to be merged with other regions for the shortest path possible.

The boolean expression for the table in Figure 22 can be minimised by optimally joining the regions that share the same class label or *don't care* label as

$$\begin{gathered} \overline{\mathrm{w}}_1 \mathrm{w}_2 \mathrm{w}_3 \vee \mathrm{w}_1 \overline{\mathrm{w}}_2 \overline{\mathrm{w}}_3 \vee \mathrm{w}_1 \overline{\mathrm{w}}_2 \mathrm{w}_3 \vee \mathrm{w}_1 \mathrm{w}_2 \overline{\mathrm{w}}_3 \\ \longrightarrow \mathrm{w}_1 \vee \overline{\mathrm{w}}_1 \mathrm{w}_2 \mathrm{w}_3 \end{gathered} \tag{16}$$

where $\vee$ denotes OR operator. The minimised expression has a smaller number of terms. Only the two terms, w_1 and $\overline{\mathrm{w}}_1 \mathrm{w}_2 \mathrm{w}_3$ are remained representing the joint regions $R_5 - R_8$ and R_4 respectively. A short tree is then built from the minimised boolean expression by placing more frequent variables at the top of the tree (see Figure 22(right)). The method for Boolean expression minimisation is close, but not suited to our problem that involves a large number of variables i.e., weak-learners.

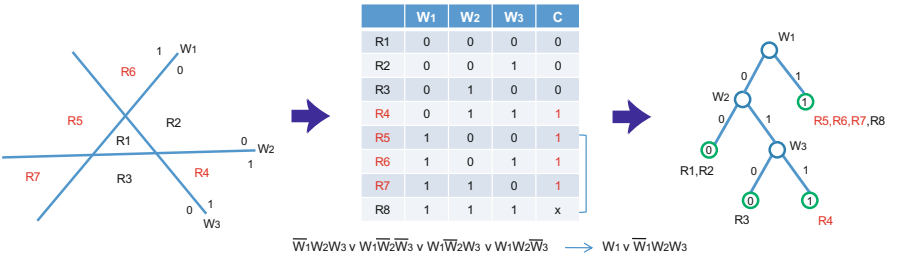


Fig. 22 Boolean expression minimisation for an optimally short tree. (a) A boosting classifier splits a space by binary weak learners (left). The regions are represented by the boolean table and the boolean expression is minimised (middle). An optimal short tree is built on the minimum expression (right).

Furthermore, all regions are treated with equal importance in the kind of methods, while an optimally short tree is learnt by considering data distribution i.e., region prior in Equation 15.

4.1.2 Growing a Super Tree

We propose a novel boolean optimisation method for obtaining a reasonably short tree for a large number of weak-learners of a boosting classifier. The classifier information is efficiently packed by using the region coding and a tree is grown by maximising the region information gain. The base algorithm is explained here, see [46] for its limitations and an improved method. The number of primitive regions 2^m is intractable when m is large. Regions R_i that are occupied by any training data points are only taken as input s.t. $p(R_i) > 0$. The number of input regions is thus smaller than the number of data points. Regions with no data points are labeled *don't care*.

Huffman coding [40] is related to our optimisation. It minimises the weighted (by region prior in our problem) path length of code (region). The technique works by creating a binary tree of nodes by maximising the entropy-based information gain. We similarly grow a tree based on the region information gain for an optimally short tree. For a certain weak-learner $w_j, j = 1, ..., m$, the regions in the left split and the right split w.r.t. the weak-learner are readily given from the boolean expressions as

$$\begin{aligned} \mathbf{R}_l &= \{R_i | B(R_i) \wedge \overline{\mathrm{w}}_1 \cdots \mathrm{w}_j \cdots \overline{\mathrm{w}}_m) = 0\} \\ \mathbf{R}_r &= \mathbf{R}_n \setminus \mathbf{R}_l \end{aligned} \tag{17}$$

where $\mathbf{R}_n$ is the set of regions arriving at the node n and $\wedge$ is AND operator. At each node, it is found the weak-learner that maximises

$$\Delta I = -\frac{\sum_{\mathbf{R}_l} p}{\sum_{\mathbf{R}_n} p} E(\mathbf{R}_l) - \frac{\sum_{\mathbf{R}_r} p}{\sum_{\mathbf{R}_n} p} E(\mathbf{R}_r) \tag{18}$$

where p is the region prior and E is the entropy function of the region class distribution, which is

Algorithm: Growing a super tree

Input: a set of data point regions R encoded by $\{B, c, p\}$
Output: a decision tree

1.Start with a root node $n = 1$ containing the list of all regions $\mathbf{R}_n$.
2.For i=1,...,m
3. Spit the node: $(\mathbf{R}_l, \mathbf{R}_r) = \text{split}(\mathbf{R}_n, \text{w}_i)$ by (17).
4. Compute the gain: $\Delta I = \text{gain}(\mathbf{R}_l, \mathbf{R}_r)$ by (18).
5.Find w_i^* that maximises the information gain.
6.If the gain is sufficient, save it as a split node. Else, save it as a leaf node.
7.Go to a child of split node and recurse the steps 2-6 setting $\mathbf{R}_n = \mathbf{R}_l$ or $\mathbf{R}_r$.

Fig. 23 Pseudocode of the algorithm.

$$Q(c^*) = \sum_{\mathbf{R}_c^*} p, \text{ where } \mathbf{R}_c^* = \{R_i | c(R_i) = c^*\}. \tag{19}$$

The node splitting is continued until all regions in a node have the coherent region label. The key ideas in the method have two-folds: 1) growing a tree from the decision regions and 2) using the region prior (data distribution). Compared to conventional decision trees built on data points, the proposed tree is grown upon smooth decision regions, guaranteeing better generalisation. Using the region prior helps getting an optimally short tree in the sense of average path length of data points. See Figure 23 for the pseudo-code of the proposed algorithm.

4.1.3 Cascade of Super Tree and Fast-Exit

Designing a cascade involves a number of parameters to set. The setting is more difficult with more stages. Our solution explained in the previous section can be seen as a convenient way of speeding up a boosting classifier up to several tens of weak-learners without need of a multi-stage cascade. We use a two stage cascade to cope with any larger number of weak-learners of a boosting classifier. It places the super tree in the first stage and the fast-exit method in the second stage. The fast-exit method, which yields the same accuracy as a boosting classifier of any number of weak-learners, is required to meet the target accuracy of a cascade. We first designed a two-stage cascade of standard boosting classifiers in a conventional way, by varying the number of weak-learners (but limiting the number of weak-learners of the first stage to less than a hundred) and the thresholds. Then, the two standard boosting classifiers were replaced with the super-tree and the fast-exit method. The proposed cascade significantly speeds up a two-stage cascade of standard boosting classifiers and the same of the fast-exits at both stages, as well as a single boosting classifier (see Section 4.2).

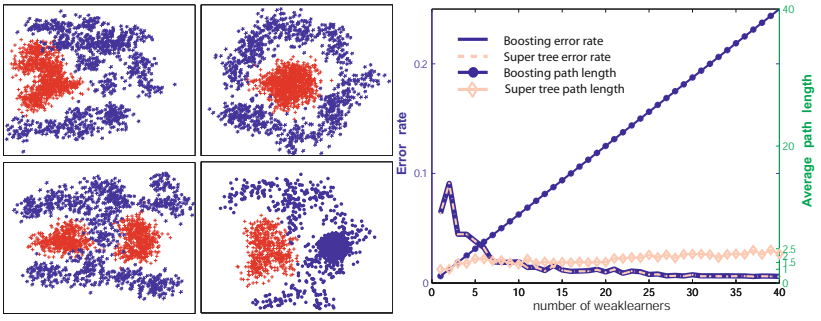


Fig. 24 Experimental results on the synthetic data. Examples of 2D synthetic data sets (left). Super tree obtains the same accuracy as the booting classifier significantly shortening the average path length (right).

4.2 Experiments and Discussions

Classification of Synthetic 2D Data

We have made twelve 2D synthetic data sets. Data points of two classes were generated from Gaussian mixtures as exemplified in Figure 24. The six test sets were created by randomly perturbing the train sets. We have compared the two methods here: a boosting classifier (AnyBoost implementation [9]) and the proposed tree using the data point regions. Vertical and horizontal lines are weak-learners of boosting. Figure 24(right) shows the results. The left and right y-axis in the graph show the classification error rate and the average path length i.e., number of weak-learners used per point respectively. Note first that the both methods do drop the error rate when the number of weak-learners is increased indicating good generalisation. The proposed method exhibited the same accuracy as the boosting classifier for all number of weak-learners. While the boosting classifier linearly increased the average path length for the number of weak-learners, the proposed method quickly converged significantly reducing down the average path length. At 40 weak-learners, the super tree speeds up the boosting classifier by 16 times.

Object Detection

For training, we used the MPEG-7 face data set [41] that has 11,845 face images collected from a few public face data sets such as Yale and XM2VTS, and non-public face data sets. BANCA face set (520 faces) and Caltech background image sets (900 images) were exploited for bootstrapping. The total number of negative-class images for training, which were either bootstrapped or randomly drawn, is 50,128. We used 21,780 Haar-like features on integral images as weak-learners. We have tested on the MIT+CMU frontal face test set [12] which consists of 130 images with 507 labeled frontal faces. The 507 face and 57000 random image patches were cropped and resized into 24x24 images. Example images are shown in

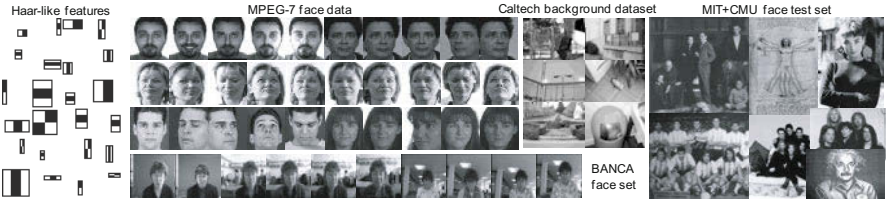


Fig. 25 Example features (weak-learners) and face images used.

No. of weak learners	Boosting			Fast exit (cascade)			Super tree (cascade)		
	False positives	False negatives	Average path length	False positives	False negatives	Average path length	False positives	False negatives	Average path length
20	501	120	20	501	120	11.70	476	122	***7.51***
40	264	126	40	264	126	23.26	231	127	***12.23***
60	222	143	60	222	143	37.24	212	142	***14.38***
100	148	146	100	148 (144)	146 (149)	69.28 (37.4)	(145)	(152)	(***15.1***)
200	120	143	200	120 (146)	143 (148)	146.19 (38.1)	(128)	(146)	(***15.8***)

Fig. 26 Experimental results on the face images. The numbers in the the brackets are for the two-stage cascades.

Figure 25. The methods compared include a standard boosting classifier, Fast exit, the cascade of two Fast exits, Super tree and the two-stage cascade of Super tree and Fast-exit. For the super tree, we used the extended regions [46]. Fixing the accuracy at 0 threshold, we have compared the average path lengths of the methods in Figure 26. For all different numbers of weak-learners, the super tree significantly reduces the average path length of the boosting classifier and the fast exit. The two-stage cascade solution of 60 weak-learner super tree and 200 weak-learner fast exit speeded up the standard boosting by 6.6-12.7 times and even the two-stage cascade of 60 and 200 weak-learner fast exits by 2.5 times. Note that the super tree exploits various combinations of weak-learners (i.e., paths) for an optimal classification speed, whereas the fast exit takes the combinations always in the order of the weak-learner weights. One can also compare the results of [35] with ours using the standard boosting and the fast-exit as proxies. Whereas the solution in [35] didn't gain much over the boosting and the fast exit method, ours significantly improved the both. More importantly, the method [35] has been tested only for 5 or 10 weak learners whereas our method in a single stage is conveniently scalable up to several tens of weak learners.

Single conventional decision trees of various pruning [37] were very poor. The best accuracy among those of the different pruned trees (false positives: 1995/false negatives: 120) is by far worse than that of the super tree of 20 weak-learners (false positives: 476/false negatives: 122). The super tree was even shorter than the decision tree: the depth of the super tree and conventional tree was about 7.5 and 9 respectively.

Fig. 27 Segmentation results. Pixels classified into the building class by Super tree (or Boosting) are shown by a darker hue.

Although the comparison has been made on the two-stage cascades, the proposed method affords a speed up over a standard multi-stage boosting cascade by replacing each stage of a boosting classifier in the cascade with a Super Tree. The speed gains over the different numbers of weak-learners in a single stage boosting classifier are reported in Figure 26. In the other sense, the proposed method can be seen as a convenient way of obtaining the comparable speed-up to a multi-stage cascade by the single super tree (or the proposed two-stage cascade).

Segmentation by pixel-wise classification

The car driving sequences [42] were exploited for the experiment. Boosting classifier and super tree were trained for the binary problem for the building class against non-building class. 1323 DCT features were drawn from 21x21 RGB image patch as weak-learners. The train set consisted of 7143 positive and 23217 negative pixels from 184 images of 11x15 pixel resolution. Randomisation in learning (similarly to [45]) reduced the train time of the boosting classifier. The test set contained 38445 points from 233 images. The correct recognition rate of Boosting of 40 weak-learners was 0.71 (as global accuracy) or 0.736 (as average class accuracy). The super tree learnt by 10 extended regions per region obtained the close accuracy as 0.70 (as global accuracy) or 0.728 (as average class accuracy) using only 15 weak-learners on average. The accuracy obtained seems comparable to [42]. Figure 27 shows the segmentation results. The blocky effect was due to the low pixel image resolution used.

Discussions

We have proposed a novel way to speed up a boosting classifier. The problem is formularised as boolean optimisation and a new optimisation method is proposed for a large number of weak-learners. The tree grown from the decision regions of a boosting classifier, called Super tree, provides many short paths and preserves the

Boosting decision regions. The single super tree delivers the close accuracy to a boosting classifier with a great speed-up for up to several tens of weak-learners. The proposed two stage cascade allows any number of weak-learners. Experiments have shown that the tree obtained is reasonably short in terms of average path length outperforming a standard boosting classifier, fast exit, their cascade. The method has been also demonstrated for segmentation problems.

5 Summary and Conclusion

We have formulated a novel discriminative co-clustering problem of images and features, and have presented the solution called MCBoost by simultaneously learning multiple boosting classifiers. Each boosting classifier in the method cooperates and competes with others, taking expertise on a subset of images. The method has been shown to yield meaningful co-clusters of images and features, significantly outperforming the conventional designs of single or multiple boosting classifiers.

The MCBoost method has been extended into a online version with a soft partitioning of the input space for object tracking. It incorporated a weighting function $\mathcal{Q}$, which ensures that coherent clusters are formed, in the MCBoost framework. Whereas existing single classifier trackers tend to adapt to a single appearance mode, forgetting previous modes, the method called MCBQ allows tracking during rapid pose changes, since it captures multiple appearances. The method can also be seen as a multi-class extension of the MIL tracker [22].

Lastly, we have proposed a novel way to convert a standard boosting classifier into a decision tree for speeding up. The conversion problem is formularised as boolean optimisation and a new optimisation method is proposed for a large number of weak-learners. The tree grown from the decision regions of a boosting classifier, called Super tree, provides many short paths (i.e., speeding up the evaluation time) and preserves the Boosting decision regions (i.e., the same accuracy). In the experiments, the super tree outperformed a standard boosting classifier, fast exit, their cascade in speed, delivering the same accuracy to that of an input boosting classifier.

Many visual recognition problems are formulated as classification problems of image sub-windows. Every possible sub-window, across pixels and scales, is evaluated to decide whether it contains an object of interest or not. The number of sub-windows is often massive, requiring the evaluation of each sub-window in a very fast manner. This study has presented required efficient classification methods. We began with a standard boosting method and have introduced largely three extensions of it to improve the performance in both accuracy and time. Major issues for future work include a more principled way to select the number of boosting classifiers in MCBoost and to form a forest of Super Trees by randomisation (similarity to Random Forests [36]). See also the discussions of each section.

Acknowledgements. Section2, Section 3 and Section 4 have been compiled from the authors' previous publications [31, 34] and [46] respectively.

References

1. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: Proc. ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, pp. 89–98 (2003)
2. Jordan, M.I., Jacobs, R.A.: Hierarchical mixture of experts and the EM algorithm. Neural Computation 6(2), 181–214 (1994)
3. Viola, P., Jones, M.: Robust real-time object detection. Int'l J. Computer Vision 57(2), 137–154 (2002)
4. Torralba, A., Murphy, K.P., Freeman, W.T.: Sharing visual features for multiclass and multiview object detection. IEEE Trans. on PAMI 29(5), 854–869 (2007)
5. Viola, P., Platt, J.C., Zhang, C.: Multiple Instance Boosting for Object Detection. In: Proc. Advances in Neural Information Processing Systems, pp. 1417–1426 (2006)
6. Li, S.Z., Zhang, Z.: Floatboost learning and statistical face detection. IEEE Trans. on PAMI 26(9), 1112–1123 (2004)
7. Sochman, J., Matas, J.: Waldboost - learning for time constrained sequential detection. Proc. CVPR 2, 150–157 (June 2005)
8. Schapire, R.: The strength of weak learnability. Machine Learning 5(2), 197–227 (1990)
9. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Boosting algorithms as gradient descent. In: Proc. Advances in Neural Information Processing Systems, pp. 512–518 (2000)
10. Sim, T., Baker, S., Bsat, M.: The CMU Pose, Illumination, and Expression Database. IEEE Trans. on PAMI 25(12), 1615–1618 (2003)
11. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: Proc. CVPR, pp. 886–893 (2005)
12. Rowley, H.A., Baluja, S., Kanade, T.: Neural Network-Based Face Detection. IEEE Trans. on PAMI 20(1), 23–38 (1998)
13. Schneiderman, H., Kanade, T.: A Statistical Model for 3D Object Detection Applied to Faces and Cars. In: Proc. CVPR (June 2000)
14. Wu, B., Nevatia, R.: Cluster Boosted Tree Classifier for Multi-View, Multi-Pose Object Detection. In: Proc. ICCV (2007)
15. Huang, C., Ai, H., Li, Y., Lao, S.: Vector Boosting for Rotation Invariant Multi-View Face Detection. In: Proc. ICCV (2005)
16. Tu, Z.: Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering. In: Proc. ICCV (2005)
17. Grossmann, E.: AdaTree: boosting a weak classifier into a decision tree. In: IEEE Workshop on Learning in Computer Vision and Pattern Recognition, p. 105 (2004)
18. Babenko, B., Dollár, P., Tu, Z., Belongie, S.: Simultaneous learning and alignment: Multi-instance and multi-pose learning. In: ECCV Workshop on Faces in Real-Life Images (2008)
19. Wojek, C., Walk, S., Schiele, B.: Multi-Cue Onboard Pedestrian Detection. In: Proc. CVPR (2009)
20. Pham, M.T., Cham, T.J.: Fast training and selection of Haar features using statistics in boosting-based face detection. In: Proc. ICCV (2007)
21. Avidan, S.: Ensemble tracking. IEEE Trans. PAMI 29(2), 261–271 (2007)
22. Babenko, B., Yang, M.-H., Belongie, S.: Visual tracking with online multiple instance learning. In: Proc. CVPR, Miami, FL (June 2009)
23. 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. 1064, pp. 329–342. Springer, Heidelberg (1996)
24. Collins, R., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. IEEE Trans. on PAMI 27(10), 1631–1643 (2005)
25. Freund, Y., Schapire, R.: A decision theoretic generalization of on-line learning and an application to boosting. J. of Computer and System Sciences 55(1), 119–139 (1997)
26. Grabner, H., Bischof, H.: On-line boosting and vision. In: Proc. CVPR, vol. 1, pp. 260–267 (2006)

27. 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)
28. Hall, P., Marshall, D., Martin, R.: Merging and splitting eigenspace models. IEEE Trans. on PAMI 22(9), 1042–1049 (2000)
29. Jebara, T., Pentland, A.: Parameterized structure from motion for 3d adaptive feedback tracking of faces. In: Proc. CVPR, pp. 144–150 (June 1997)
30. Jones, M., Viola, P.: Fast multi-view face detection. Technical Report 96, MERL (2003)
31. Kim, T.-K., Cipolla, R.: MCBoost: Multiple classifier boosting for perceptual co-clustering of images and visual features. In: Proc. Advances in Neural Information Processing Systems, Vancouver, Canada (December 2008)
32. Lee, K.-C., Ho, J., Yang, M.-H., Kriegman, D.: Visual tracking and recognition using probabilistic appearance manifolds. Computer Vision and Image Understanding 99(3), 303–331 (2005)
33. Avidan, S.: SpatialBoost: Adding Spatial Reasoning to AdaBoost. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 386–396. Springer, Heidelberg (2006)
34. Kim, T.-K., Woodley, T., Stenger, B., Cipolla, R.: Online Multiple Classifier Boosting for Object Tracking. In: Proc. of IEEE CVPR Workshop on Online Learning for Computer Vision, San Francisco, USA (June 2010)
35. Zhou, S.: A binary decision tree implementation of a boosted strong classifier. In: IEEE Workshop on Analysis and Modeling of Faces and Gestures, pp. 198–212 (2005)
36. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
37. Quinlan, J.: Bagging, boosting, and c4.5. In: Proc. National. Conf. on Artificial Intelligence, pp. 725–730 (1996)
38. Schwender, H.: Minimization of Boolean Expressions Using Matrix Algebra, Technical report, Collaborative Research Center SFB 475. University of Dortmund (2007)
39. Chen, J.: Application of Boolean expression minimization to learning via hierarchical generalization. In: Proc. ACM Symposium on Applied Computing, pp. 303–307 (1994)
40. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms. MIT Press and McGraw-Hill (2001)
41. Kim, T.-K., Kim, H., Hwang, W., Kittler, J.: Component-based LDA Face Description for Image Retrieval and MPEG-7 Standardisation. Image and Vision Computing 23(7), 631–642 (2005)
42. Brostow, G.J., Shotton, J., Fauqueur, J., Cipolla, R.: Segmentation and Recognition Using Structure from Motion Point Clouds. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 44–57. Springer, Heidelberg (2008)
43. Basak, J.: Online adaptive decision trees. Journal of Neural Computation 16, 1959–1981 (2004)
44. Yeh, T., Lee, J., Darrell, T.: Adaptive Vocabulary Forests for Dynamic Indexing and Category Learning. In: Proc. ICCV (2007)
45. Rahimi, A., Recht, B.: Random Kitchen Sinks: Replacing Optimization with Randomization in Learning. In: Proc. Neural Information Processing Systems (2008)
46. Kim, T.-K., Budvytis, I., Cipolla, R.: Making a Shallow Network Deep: Growing a Tree from Decision Regions of a Boosting Classifier. In: Proc. of British Machine Vision Conference, Aberystwyth, UK (2010)

Simultaneous Detection and Tracking with Multiple Cameras

Murtaza Taj and Andrea Cavallaro

1 Introduction

Tracking targets using multiple cameras is an important processing step for applications such as sports analysis, traffic monitoring, behavior detection and event recognition. The multi-camera tracking problem has been mostly addressed in the literature as detection-based tracking: objects of interest (targets) are first detected and then associated over time [1]. Data from different cameras can be combined either after tracking (in *track-first* approaches) or before tracking (in *fuse-first* approaches).

Track-first approaches localize and track objects in each camera view [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. Then the tracks are projected onto a common fusion space (usually a common view or ground plane) to generate extended tracks across the camera network. Track-first approaches solve the correspondence problem in each camera view as well as on the fusion space.

Fuse-first approaches combine the information from multiple cameras prior to tracking [13, 14, 15, 16]. The fused data can be either the projection of raw pixels from each view or of foreground pixels only [17] and is referred to as detection volume or occupancy mask. In general, a detection step is required to localize the targets in the detection volume prior to tracking. Alternatively, simultaneous detection and tracking can be performed that does not require an explicit detection step [18, 19, 20]. In this case, the evidence of existence of a track over time is used to declare the presence of a target. These strategies are also referred in literature as *track-before-detect* approaches [21] (Fig. 1) and will be the main focus of this chapter.

Murtaza Taj · Andrea Cavallaro
Queen Mary University of London, UK
e-mail: {murtaza.taj,andrea.cavallaro}@elec.qmul.ac.uk

R. Cipolla et al. (Eds.): Machine Learning for Computer Vision, SCI 411, pp. 197–214.
springerlink.com

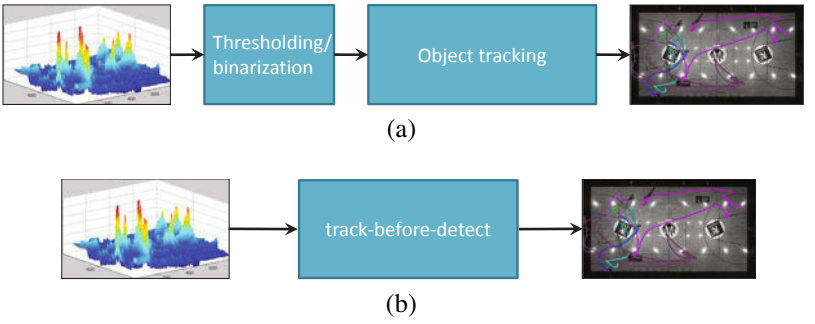


Fig. 1 (a) Detection-based tracking (b) Simultaneous detection and tracking via track-before-detect.

2 Multi-camera Tracking: A Brief Overview

Track-first multi-camera trackers fuse trajectory information from each cameras on a reference-view [10, 8] or on a top view of the scene [2, 3, 7, 9]. The tracking process in each camera may be working independently [10, 2] or in collaboration with the tracking processes in other cameras [3, 9]. Non-collaborative tracking can be performed using a multi-target tracker based on graph matching on each camera, followed by fusion on the top view using feature clustering [2]. Candidate tracks are projected back on the image view for further validation. In collaborative tracking, track estimates in one view are used as part of the measurement in another view [3, 7, 9]. In [3], targets are first tracked using a particle filter in each view. Then the particles and the principle axis lines of each target are projected onto the top view. The intersection of these lines is used as the target locations on the top view. The particles in each view are sampled from camera-view particles and from top-view particles. Similarly, in [9], multiple independent regular particle filters (MIPFs) are used to track each target in each camera view. The posterior in one camera is computed by using the measurements from all the cameras. Likewise, in [7] the 2D estimates of the target state from each camera view are projected onto the top view. These estimates are then used as observations for a Gaussian Mixture Probability Hypothesis Density (GMPHD) filter for 3D tracking on the top view.

Fuse-first multi-camera trackers perform detection and tracking on the fused data [13, 14, 15, 16]. Similar to [3], in [16] the vertical axes of a target across views are mapped on the top-view plane and their intersection point on the ground is computed to obtain the feet location of a person (target). These top-view feet locations are then tracked using a particle filter. Projecting only feet locations make this approach sensitive to detection errors in camera views and inapplicable in crowded scenarios where feet locations may not be visible. To avoid this problem, in [14] the foreground mask from each camera view is projected onto the top view to obtain an occupancy map. The tracking of each object is performed using the Viterbi algorithm. This approach can only process a batch of N frames at a time thus introducing latency in the generation of the results.

Simultaneous detection and tracking can be performed via *track-before-detect* (TBD), an approach that considers the input signal 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 [22] or by employing non-linear state estimation techniques, such as particle filtering (PF) [23], which are computationally less expensive. A recursive Bayesian single-target TBD is proposed in [24] using PF. This method assumes a point target and extends the target state with the signal intensity, based on the assumption that the intensity related to the target is unknown. This results in filtering components belonging to the noise only. To further improve robustness to noise, a gradual change in number of targets can be imposed by extending the target state with an existence variable and the use of a jump Markov model [25]. Examples of adaptation of the TBD algorithm to real applications include multi-target multi-microphone tracking algorithm applied to audio [26], tracking in single sensor infra-red sequences and multi-target multi-camera tracking [20].

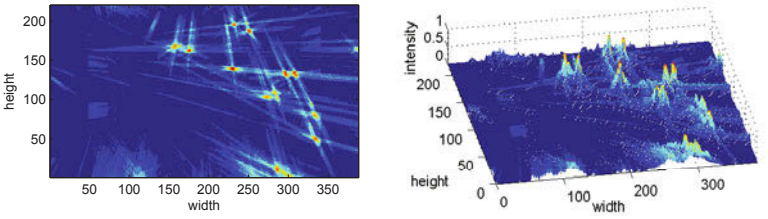
A summary of the state-of-the-art multi-camera tracking approaches is shown in Table 1, whereas more detailed discussions can be found in [27, 28].

Table 1 Multi-camera track-first and fuse-first algorithms. (Key: GMPHD: Gaussian Mixture Probability Hypothesis Density filter; BT: Bayes tracker; KF: Kalman filter; PF: Particle filter; GM: Graph matching; CFI: Caratheodory-Fejer Interpolation; VA: Viterbi algorithm; MGC: Minimum graph cut; M: Manual; A: Automatic)

	Ref.	Features	Algo.	Calib.	Multi-target
track-first	[2]	2D position, size, velocity	GM	M	Yes
	[3]	2D position, size	PF	M	No
	[4]	position, velocity, size and color	any	A	NA
	[6]	pixels, manifold learning	CFI	No	Yes
	[7]	position, size and color histogram	GMPHD	M	Yes
	[8]	2D position, size, velocity	KF	M	No
	[9]	5D state space using ellipses	PF	M	No
	[10]	2D position, height and intensity	BT	M	No
	[11]	field–of–view lines	any	A	NA
fuse-first	[13]	multiple planes occupancy map	MGC	A	Yes
	[14]	color and motion	VA	M	Yes
	[15]	head position	BT	M	Yes
	[16]	vertical axis of the target, ground position	PF	M	Yes
	[20]	signal intensity	PF	M	Yes

3 Multi-camera Detection Volume

Let a wide area be monitored by a set $C = \{C_1, \cdots, C_N\}$ of N cameras. Prior to performing multi-camera tracking, let the foreground from each camera c^{th} be projected on the top view π. This transformation can be performed through a projection matrix computed using corresponding points [29]. To further improve the effectiveness of



(a) (b)

Fig. 2 Example of multi-camera detection volume: (a) 2D view; (b) 3D view.

(a) (b)

Fig. 3 Example of parallax errors. (a) Crop of a camera view with 3 targets. (b) Corresponding detection volume showing 3 high intensity regions (the targets) and a 4^{th} region generated by parallax error.

tracking in the fused domain, a multi-level homography is used in [30, 13]. The correspondence between multiple views is established automatically using SIFT features, followed by RANSAC to reject outliers. This procedure generates a planar homography which is then used to compute multi-level homographies along the vertical vanishing points. This results in multiple projection planes that are parallel to the top-view. These projections on multiple planes can be treated separately to obtain the information about the shape of the target [13] or can be collapsed to obtain a detection volume (Fig. 2).

The signal intensity at each position in the detection volume is proportional to the number of foreground pixels being projected on that position. By using a multi-level homography, the pixels representing different portions of a person in the image view along the vertical axis (feet, legs, torso, neck, head) are projected around the same position on the top view, thus increasing the signal intensity around the feet location: each object therefore occupies multiple pixels on the top view (Fig. 2(a)). The signal strength also depends on the number of cameras observing a region, as this results in points contributed from multiple cameras being projected on the same location on the top view. However, when an object is projected on the plane, also

pixels that do not belong to that plane are projected incorrectly creating a *shadow* of the object along the plane. These projected shadows from multiple targets can overlap with each other and create false signal intensities in the detection volume. These noise components are referred to as *parallax errors* (Fig. 3), which have to be accounted for by simultaneous detection and tracking approaches.

4 Track-Before-Detect on the Detection Volume

Prior to discussing the multi-target formulation of simultaneous detection and tracking on the detection volume, let us first introduce the single-target track-before-detect formulation based on particle filtering [24].

4.1 Single Target Track-Before-Detect

Let $\mathbf{x}_k$ be the target state vector at time k, using a discrete time model with a fixed sampling period τ. The state can be defined as

$$\mathbf{x}_k = (x_k, \dot{x}_k, y_k, \dot{y}_k, I_k)^T, \tag{1}$$

where (x_k, y_k) are the position components, $(\dot{x_k}, \dot{y_k})$ are the velocity components and I_k is the value of the target signal strength (intensity) at time k at position (x_k, y_k). As mentioned in the previous section, the signal intensity is determined, when using foreground masks, by the number of foreground pixels being projected in a specific position. The state evolution can be modeled as

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathcal{N}_k^p), \tag{2}$$

where $f(.)$ is the state transition function and $\mathcal{N}_k^p$ is the process noise. For a linear stochastic process, the state evolution can be expressed as

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathcal{N}_k^p, \tag{3}$$

where $\mathbf{F}$ is the state transition matrix, defined as

$$\mathbf{F} = \begin{bmatrix} B & 0_{2\times 2} & 0_{2\times 1} \\ 0_{2\times 2} & B & 0_{2\times 1} \\ 0_{1\times 2} & 0_{1\times 2} & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} \tag{4}$$

where $0_{m\times n}$ denotes an $m \times n$ matrix of zeros and τ is the sampling interval. The process noise $\mathcal{N}_k^p$ models the disturbances affecting the target state and is generally modeled as a zero mean Gaussian random variable [21] with covariance $\mathbf{Q}$, defined as

$$\mathbf{Q} = \begin{bmatrix} \mathbf{D} & 0_{2\times 2} & 0_{2\times 1} \\ 0_{2\times 2} & \mathbf{D} & 0_{2\times 1} \\ 0_{1\times 2} & 0_{1\times 2} & q_2\tau \end{bmatrix}, \mathbf{D} = \begin{bmatrix} \frac{q_1}{3}\tau^3 & \frac{q_1}{3}\tau^2 \\ \frac{q_1}{2}\tau^2 & q_1\tau \end{bmatrix}, \tag{5}$$

where q_1 and q_2 are the process noise in target motion and intensity.

Let $\mathbf{z}_k = \{z_k(i,j) : i = 1,\cdots W, j = 1,\cdots,H\}$ be the measurement, at each time k, encoded in a $W \times H$ resolution image. At each pixel position, the measurement intensity $z_k(i,j)$ is either due to the presence of the target or due to measurement noise $\mathcal{N}_k^m$; that is

$$z_k(i,j) = \begin{cases} h_k(i,j)(\mathbf{x}_k) + \mathcal{N}_k^m(i,j) & \text{if target is present} \\ \mathcal{N}_k^m(i,j) & \text{if target is not present} \end{cases}. \tag{6}$$

$\mathcal{N}_k^m$ is modeled as a zero mean Gaussian sequence which is assumed to be mutually independent from the process noise. $h_k(i,j)(.)$ is the contribution of the target intensity at pixel position (i,j). In the case of a point target, the distribution of the target intensity over the surrounding pixels will be only due to the sensor point spread function and can be approximated as [21]

$$h_k(i,j)(\mathbf{x}_k) \approx \frac{\Delta_x \Delta_y I_k}{2\pi A^2} \exp\left(-\frac{(i\Delta_x - x_k)^2 + (j\Delta_y - y_k)^2}{2A^2}\right), \tag{7}$$

where A models the amount of blurring introduced by the sensor and $\Delta_x \times \Delta_y$ is the size in pixels of the segment centered at $(i\Delta_x, j\Delta_y)$. This indicates that each target occupies multiple pixels in the measurement $\mathbf{z}_k$, instead of being a point target (Fig. 2(a-b)).

Given the set of measurements $Z_k = \{\mathbf{z}_m | m = 1,\cdots,k\}$ up to time k, the objective is to recursively quantify some degree of belief in the state $\mathbf{x}_k$ taking different values, i.e., to estimate the posterior *pdf* $p(\mathbf{x}_k|Z_k)$. Using the Bayesian recursion, the posterior *pdf* $p(\mathbf{x}_k|Z_k)$ can be computed in two steps: *prediction* and *update*. In the *prediction* step, the prior density of the state at time k is obtained using the Chapman-Kolmogorov equation:

$$p(\mathbf{x}_k|Z_{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|Z_{k-1}) d\mathbf{x}_{k-1}, \tag{8}$$

where $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the transition density defined by the target model (Eq. 2) and $p(\mathbf{x}_{k-1}|Z_{k-1})$ is the posterior at time $k-1$. The *update* step is carried out using the measurement at time k by applying Bayes' rule:

$$p(\mathbf{x}_k|Z_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|Z_{k-1})}{\int p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|Z_{k-1}) d\mathbf{x}_k}, \tag{9}$$

where $p(\mathbf{z}_k|\mathbf{x}_k)$ is the likelihood function.

The above algorithm can be implemented using a Sampling Importance Resampling (SIR) particle filter [23] where a posterior density is represented by a set of particles each with associated weight $\{\omega_k^n, \mathbf{x}_k^n\}$.

4.2 Track-Before-Detect Particle Filter

Particle filtering approximates the posterior density with a set of particles. In the *prediction* step, two sets of particles are drawn to estimate the predicted density, namely *new-born* particles and *surviving* particles. The new-born particles are the set of J_k particles for which the target state is drawn as a sample from a proposal distribution $p(\mathbf{x}_k|Z_k)$. The proposal distribution $p(\mathbf{x}_k|Z_k)$ could be any appropriate distribution, such as a uniform distribution where at each position (x_k, y_k) in the measurement $\mathbf{z}_k$, equal number of particles are drawn. Such a distribution is appropriate when the signal-to-noise ratio (SNR) is very low. In case of moderate or high SNR, the proposal distribution $p(\mathbf{x}_k|Z_k)$ can be the measurement $\mathbf{z}_k$ itself, normalized between zero and 1 such that at each position (x_k, y_k) in the measurement $\mathbf{z}_k$ the number of particles drawn is proportional to the signal intensity $I_k(x_k, y_k)$ (Fig. 2(a-b)). The surviving particles are the set of L_{k-1} particles that continue to stay alive. These particles are generated from the proposal density $q_k(\mathbf{x}_k|\mathbf{x}_{k-1}, Z_k)$ based on the target dynamic model such that the current state of each of the surviving particles is estimated by applying Eq. 3 (Fig. 8(b)).

Particle filtering approximates the densities $p(\mathbf{x}_k|Z_k)$ with a sum of $L_{k-1} + J_k$ Dirac functions centered in $\{\mathbf{x}_k^n\}_{n=1,\ldots,L_{k-1}+J_k}$ as

$$p(\mathbf{x}_k|Z_k) \approx \sum_{n=1}^{L_{k-1}+J_k} \omega_k^n \delta(\mathbf{x}_k - \mathbf{x}_k^n), \tag{10}$$

where ω_k^n are the weights associated with the particles. The weights are calculated in [23] as

$$\omega_k^n \propto \omega_{k-1}^n \frac{p(\mathbf{z}_k|\mathbf{x}_k^n) p(\mathbf{x}_k^n|\mathbf{x}_{k-1}^n)}{q(\mathbf{x}_k^n|\mathbf{x}_{k-1}^n, \mathbf{z}_k)}. \tag{11}$$

$q(.)$ is the importance density function. When $q(.) = p(\mathbf{x}_k|\mathbf{x}_{k-1}^n)$ (i.e., the transitional prior), then

$$\omega_k^n \propto \omega_{k-1}^n p(\mathbf{z}_k|\mathbf{x}_{k-1}^n). \tag{12}$$

In the *update* step, for each pixel (i, j), the likelihood $p(z_k(i, j)|\mathbf{x}_k^n)$, for the combined set of $L_{k-1} + J_k$ particles is computed. Given the sensor model defined in Eq. 6, the likelihood function can be expressed as [24]

$$p(\mathbf{z}_k|\mathbf{x}_k) = \begin{cases} \prod_{i=1} \prod_{j=1} p_{S+\mathcal{N}}(z_k(i, j)|\mathbf{x}_k^n) & \text{if a target is present} \\ p_{\mathcal{N}}(z_k(i, j)) & \text{if a target is not present} \end{cases}, \tag{13}$$

where $p_{\mathcal{N}}(z_k(i, j))$ is the *pdf* of the background noise in pixel (i, j) and $p_{S+\mathcal{N}}(z_k(i, j)|\mathbf{x}_k^n)$ is the likelihood of the target signal affected by noise in pixel (i, j). The product between the *pdf* values computed for each pixel (i, j) is based on the assumption that the measurement noise $\mathcal{N}_k^m(i, j)$ is independent from pixel to pixel.

The final likelihood is obtained by taking the likelihood ratio in pixel (i, j) for a target in state $\mathbf{x}_k^n$ as

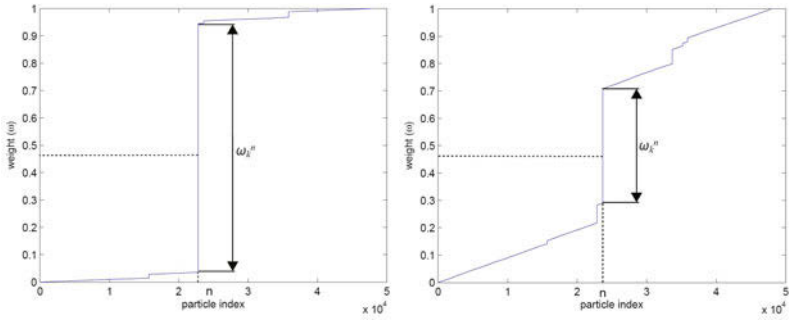


Fig. 4 Sample cumulative of particle weights. (a) Degeneracy problem showing all but one particle having negligible normalized weights. (b) The n^{th} particle $\mathbf{x}_k^n$ has higher chances of being selected due to having high weight ω_k^n.

$$p(z_k(i,j)|\mathbf{x}_k^n) = \frac{p_{S+\mathcal{N}}(z_k(i,j)|\mathbf{x}_k^n)}{p_{\mathcal{N}}(z_k(i,j))} = \exp\left(-\frac{h_k(i,j)(\mathbf{x}_k)(h_k(i,j)(\mathbf{x}_k) - 2z_k(i,j))}{2A^2}\right). \tag{14}$$

Since the pixels are assumed to be conditionally independent, the likelihood of the whole image is computed by taking the product over the pixels; thus, the updated particle weights are computed as

$$\tilde{\omega}_{k|k-1}^n = \prod_{i=w_i(\mathbf{x}_{k|k-1}^n)} \prod_{j=w_j(\mathbf{x}_{k|k-1}^n)} p(z_k(i,j)|\mathbf{x}_k^n), \tag{15}$$

where $w_i(.)$ and $w_j(.)$ indicates that only the pixels affected by the target are used in the likelihood computation which are selected by using a fixed size window. The weights are finally normalized with the sum of all weights $\Omega_k = \sum_{n=1}^{L_{k-1}+J_k} \omega_{k|k-1}^n$ as

$$\omega_{k|k-1}^n = \frac{\tilde{\omega}_{k|k-1}^n}{\Omega_k}. \tag{16}$$

The variance of these importance weights $\omega_{k|k-1}$ can only increase over time [31]. This means that after certain number of particle filtering steps, all but one particle will have negligible normalized weights (Fig. 4(a)). This phenomenon is called the degeneracy problem [21]. To avoid the degeneracy of particles, resampling is applied which eliminates samples with low importance weights and replicates samples with high importance weights by using the cumulative sum of particle weights (Fig. 4(b)). The combined set of $L_{k-1} + J_k$ particles are resampled to reduce the number to L_k only by selecting particles for which $\omega_k^n > \lambda_\omega$, where λ_ω is the minimum allowed particle weight. This process involves generating L_k random variable from the uniform distribution on the interval $[0,\ 1]$. For each of the L_k values, a

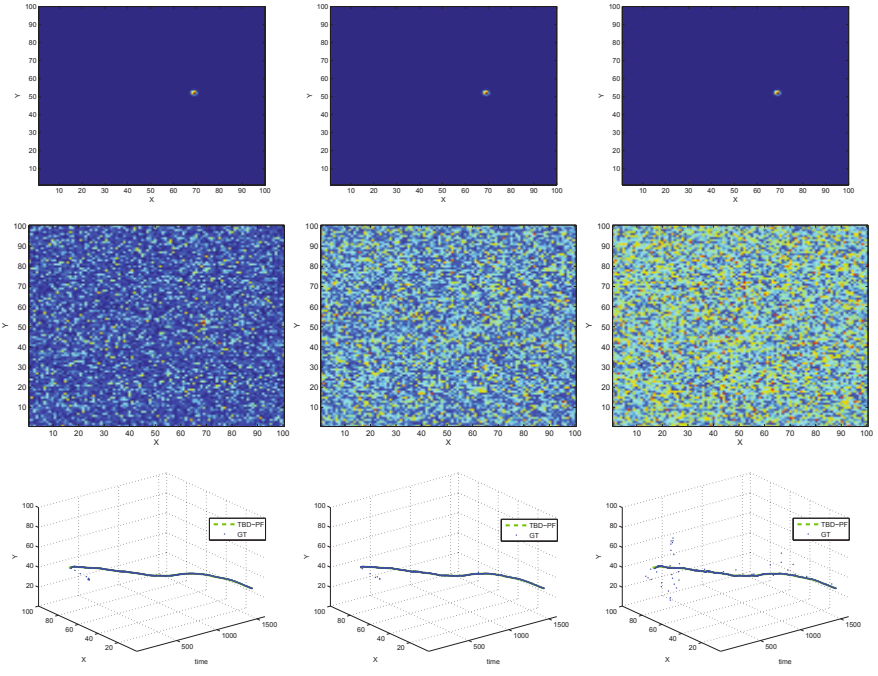


Fig. 5 Sample single target track-before-detect results with varying SNR values: (Row 1) sample frames from input data without noise indicating a target with hotter values. (Row 2) sample frames from input data with noise illustrating that target is difficult to detect by visual inspection at low SNR. (Row 3) Tracking results (mean particle position for every k). (a) SNR = 18.3422dB, (b) SNR = 8.6969dB and (c) SNR = 6.2613dB. (Blue dots: estimated positions; green dashes: ground truth).

particle whose weight corresponds to that value is propagated. The resampled particles weights are set to $\omega_{k-1}^n = 1/(L_{k-1} + J_k)\ \forall n$. This means there is no need to pass on the importance weights from one time step to the next and Eq. 12 can be simplified to

$$\omega_k^n \propto p(\mathbf{z}_k|\mathbf{x}_{k-1}^n). \tag{17}$$

That is, the weights are proportional to the likelihood function.

Figure 5 shows an example of single target track-before-detect particle filter using three different SNR values of synthetic data. The synthetic data is generated by computing a target track using a motion model. This track is then converted into an input image of resolution WxH where the position of the target is represented by a Gaussian with standard deviation of 2 pixels (Fig. 5 (Row 1)). White Gaussian noise is then added on this image multiple times to achieve a signal with different SNR values (Fig. 5 (Row 2)). Although with SNR= 8.6969dB and SNR= 6.2613dB the target cannot be observed visually due the noise (Fig. 5 (Row 2)(b-c)), it was

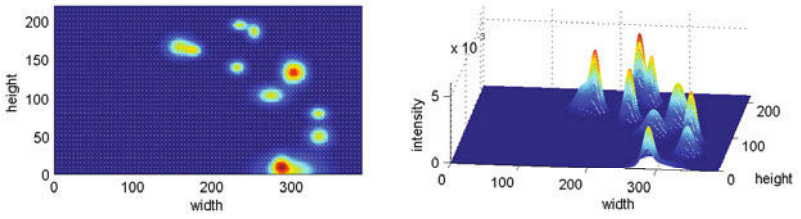


Fig. 6 Examples of 2D and 3D visualizations of the detection volume where measurements at each position corresponding to a target can receive contributions also from other targets.

correctly tracked (Fig. 5 (Row 3)(b-c)). When SNR= 6.2613dB, the algorithm had some difficulties in identifying the target location; however, once enough particles were drawn around the target, it was tracked consistently.

4.3 Multi-camera, Multi-target Track-Before-Detect

In the case of multiple targets, the measurement at each pixel (i, j) can have a contribution from all the targets (Fig. 6) and Eq. 6 can be modified to

$$z_k(i,j) = \begin{cases} \sum_{t=1}^{N_k^O} h_k(i,j)(\mathbf{x}_k^t) + \mathcal{N}_k(i,j) & \text{if } N_k^O \text{ targets present at time } k \\ \mathcal{N}_k(i,j) & \text{if no target present} \end{cases}. \quad (18)$$

The approximation shown in Eq. 7 is based on a point target assumption and is a truncated 2D Gaussian density with circular symmetry. A similar approximation can be used in the case of multiple targets by tuning the values for Δ_x, Δ_y and A, respectively. This approach aims at filtering the noise due to the parallax error (see Fig. 3).

A block diagram of the multi-target track-before-detect particle filtering is shown in Fig. 7 and its details are discussed below.

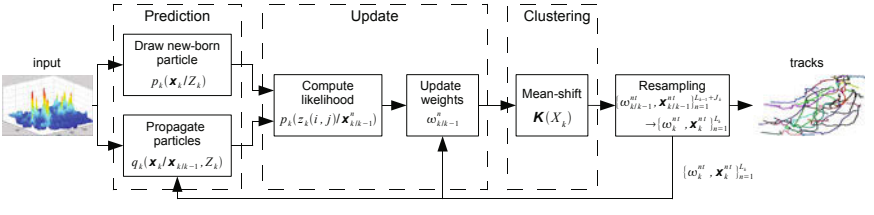


Fig. 7 Block diagram of the multi-target track-before-detect particle filter approach.

4.3.1 Prediction and Update

The prediction step remains the same as in the case of single targets. If all targets follow the same motion model, this prediction step is correct as each particle contains the velocity components $(\dot{x}_k, \dot{y}_k)$ of the target it represents. Tracking targets with a different dynamic model can be performed by incorporating Interacting Multiple Models (IMM) [32].

As different targets may have different intensity levels and in TBD the weight update is a function of the target intensity, this results in lower weight assignment to weaker targets. To address this issue each target can be considered individually in the update step and Eq. 15 can be re-written for multiple targets TBD as

$$\tilde{\omega}^{nt}_{k|k-1} = \prod_{i \in w_i(\mathbf{x}^{nt}_{k|k-1})} \prod_{j \in w_j(\mathbf{x}^{nt}_{k|k-1})} p(z_k(i,j)|\mathbf{x}^n_k), \tag{19}$$

where $\mathbf{x}^{nt}_{k|k-1}$ is the n^{th} particle at time k belonging to the t^{th} target. The weights are normalized with the sum of all weights associated to t^{th} target $\Omega^t_k = \sum_{n \in t} \omega^{nt}_{k|k-1}$ as

$$\omega^{nt}_{k|k-1} = \frac{\tilde{\omega}^{nt}_{k|k-1}}{\Omega_k \Omega^t_k}. \tag{20}$$

Here the component Ω_k is used to further normalize the weights so that they lie between 0 and 1. This is used instead of the number of targets as there are some particles generated using another proposal density $p(\mathbf{x}_k|Z_k)$.

4.3.2 Clustering

The particle filter may perform poorly when the posterior is multi-modal as the result of the presence of multiple targets [33]. To solve this problem, an existence variable and the jump Markov model [18, 25] can be used. However, this solution requires that the total number of targets is known a priori. Alternatively, clustering of the particles can be employed for which both parametric and non-parametric approaches exist and the aforementioned limitation can be eliminated.

After the update step, the particles are clustered using mean-shift for the association of an identity with each particle. Mean-shift clustering climbs the gradient of a probability distribution to find the nearest dominant mode or peak [34]. Mean-shift is preferred here as it is a non-parametric clustering technique that does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters.

Given $L_{k-1}+J_k$ particles $\{\mathbf{x}_k^n, n=1,\cdots,L_{k-1}+J_k\}$ on a 2-dimensional space $\mathbb{R}^2$ using (x_k, y_k) only, the multivariate kernel density estimate obtained with kernel $\mathcal{K}(\mathbf{x})$ and bandwidth h is

$$f(\mathbf{x}_k) = \frac{1}{(L_{k-1}+J_k)h^2} \sum_{n=1}^{L_{k-1}+J_k} \mathcal{K}\left(\frac{X_k - \mathbf{x}_k^n}{h}\right). \tag{21}$$

The bandwidth h is set as $h = 2q_1$ based on the target covariance Q (see Eq. 5). The mean-shift algorithm tends to maximize the density whose modes are located at the zeros of the gradient $\bigtriangledown f(\mathbf{x}_k)$.

As this clustering may generate more clusters than targets in the scene, a cluster merging process is performed to fuse similar clusters. The fusion can be based on several criterias such as cluster population, density, mean and covariance. Finally, an identity is assigned to each particle based on its cluster membership. If all the particles in a cluster are new-born, they can be issued a new identity; otherwise all cluster members can be assigned the identity with the highest population within the cluster.

Figure 8(c) shows the particles before clustering, whereas the clustered particles are shown in Fig. 8(d-f). In Fig. 8(d-f) each color indicates a unique cluster and particles colored in dark blue in Fig. 8(d) are the pruned particles.

4.3.3 Resampling

Similar to the single target case, to avoid the degeneracy problem [23] particles can be resampled. Resampling is performed according to the particle weights. Here again the single target resampling strategy based on the cumulative distribution function *cpdf* of particle weights will not work as it is insensitive to the particle location. Particles with lower weights (such as those associated to new-born targets) will not be able to have enough representation in the mixture distribution.

As shown in Fig. 9, ω_k^{n1}, ω_k^{n2} and ω_k^{n3} are the weights for a particle representing the state of target 1, 2 and 3, respectively. It can also be seen that $\omega_k^{n1} > \omega_k^{n2} > \omega_k^{n3}$ and hence different number of particles will be generated from their corresponding particles. This process would result in an unfair resampling where more particles will be used to represent the state of a particular target because of its particle weights. This will create a hindrance in initializing new tracks in the presence of existing targets. To overcome this problem, the resampling can be performed individually for each cluster: only the weights of the particles that are associated with the cluster can be used to create a cumulative distribution function *cpdf*. The resampling can then be performed for each cluster individually as for the single target case (Fig. 4(b)).

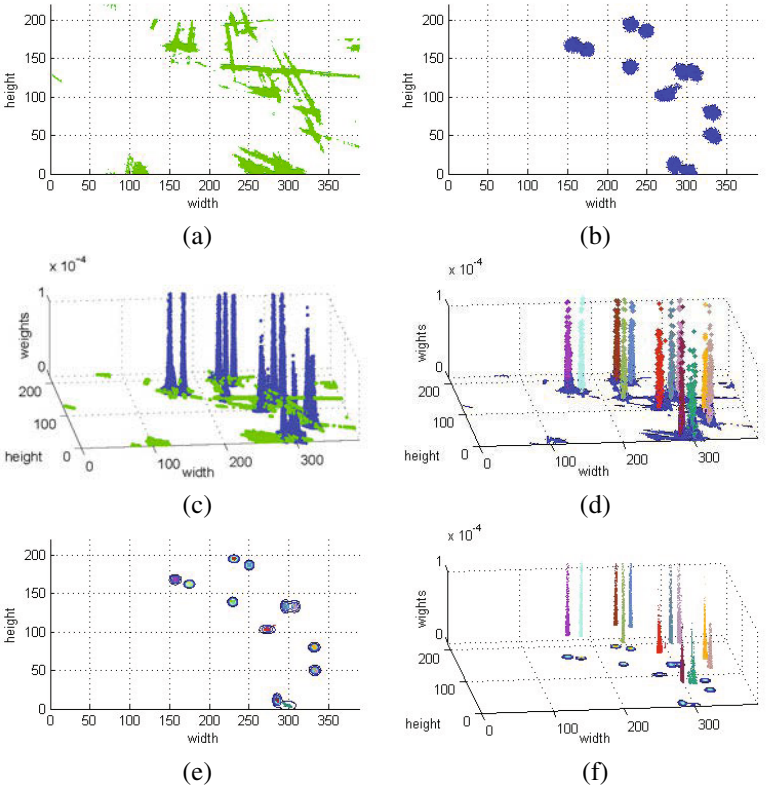


Fig. 8 Sample results at intermediate steps of the multi-target track-before-detect particle filtering. (a-b) Output at prediction step showing new-born and propagated particles. (c) Weight assignment at update step. (d) Uniquely color-coded clusters of particles corresponding to targets and pruned particles (dark blue color). (e-f) Uniquely color-coded clusters of particles.

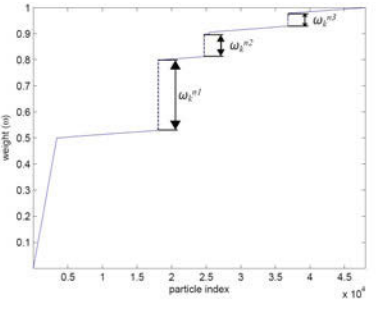


Fig. 9 Sample cumulative particle weights. The weights are associated with particles belonging to different targets (ω_k^{n1}, ω_k^{n2} and ω_k^{n3}).

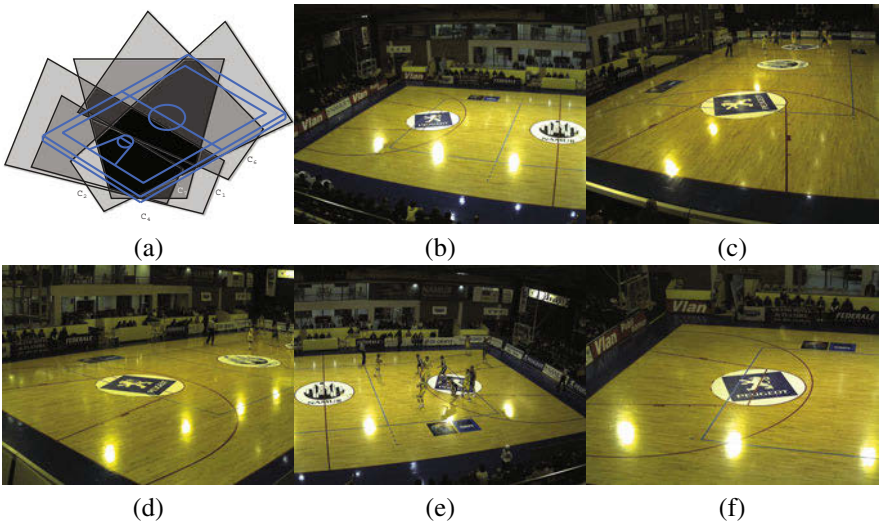


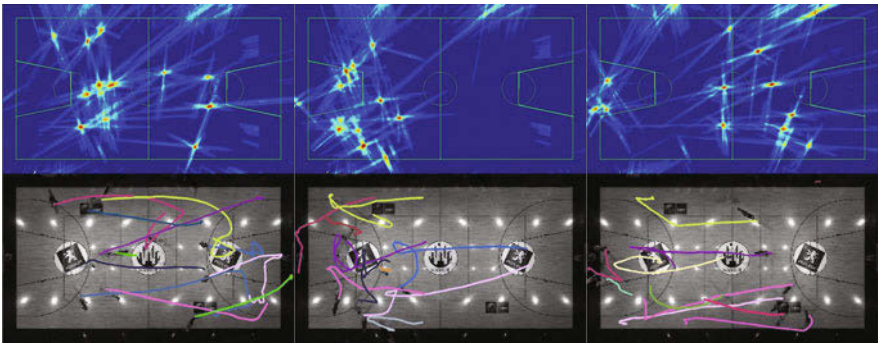
Fig. 10 Multi-camera view of a scene: (a) configuration of cameras covering a basketball court (excluding two top-mounted cameras). (b) View of camera 1. (c) View of camera 2. (d) View of camera 4. (e) View of camera 6. (f) View of camera 7.

4.4 Discussion

The evaluation of the multi-target track-before-detect particle filter (MT-TBD-PF) is performed on the APIDIS dataset, which consists of a basketball match scenario captured using five partially overlapping cameras (Fig. 10(b-f)) and two top-mounted with fish eye lenses. There are in total 12 targets in the video (10 players and 2 referees). The players have similar appearances and are difficult to distinguish from the background. Two types of detection volume are generated: one with 5 contributing cameras (excluding the fish eye lens cameras) and one with contributing 7 cameras. The goal is to quantify the difference in performance when not using the top-mounted cameras, as this camera positioning is generally not available.

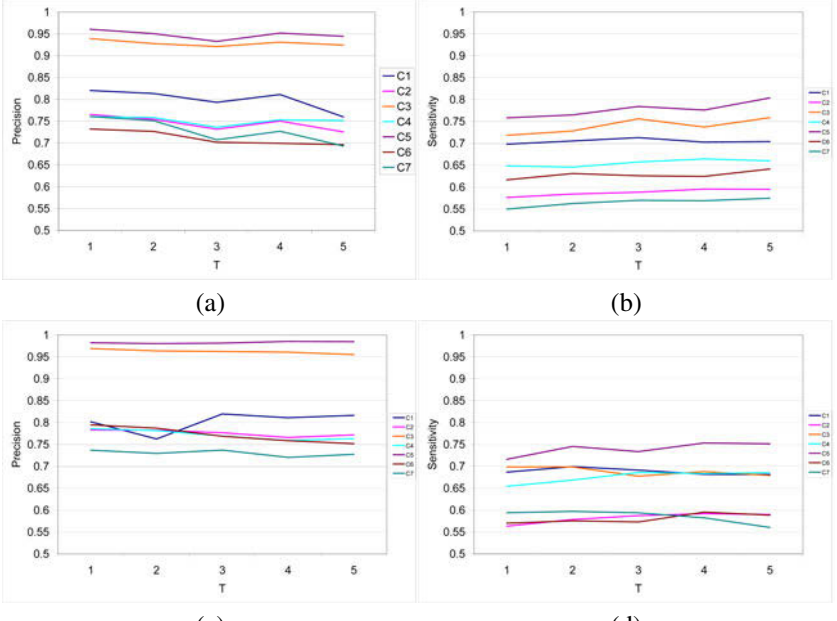
The parameters used in the experiments are as follows. The minimum allowed target weight is $\lambda_\omega = 10^{-5}$. The bandwidth chosen for Mean Shift (MS) is $h = 5$, which is appropriate for clustering particles generated around a target that is affected by a blurring with $A = 0.3733$ and $(\Delta_x, \Delta_y) = (1,1)$. The σ value for the likelihood computation was set to 0.3. The tracking is done using 3000 particles per target.

A visualization of the results from the proposed approach is given in Fig. 11 (Row 2). The projection of the detection mask on the top view using the multi-layer homography is in Fig. 11 (Row 1). Several challenges regarding the data can be observed such as parallax error and different amounts of noisy intensity values in different regions. The tracks generated on the top view are first reprojected onto each camera view and then evaluated against the ground truth. Precision (Fig. 12 (a,c)) and sensitivity (Fig. 12 (b,d)) are computed for results obtained from both the



(a) (b) (c)

Fig. 11 Sample fusion and multi-target tracking results on the top view for the APIDIS dataset. (Row 1) Detection volume on the top view. (Row 2) Tracking results obtained with the multi-camera track-before-detect particle filtering approach.



(a) (b)

(c) (d)

Fig. 12 Precision and sensitivity scores for cameras C1-C7 of the APIDIS dataset. (a-b) Precision and sensitivity of tracking results generated using detection volume obtained from 5 cameras. (c-d) Precision and sensitivity of tracking results generated using a detection volume obtained from 7 cameras.

5-camera detection volume and the 7-camera detection volume, at 5 different sampling intervals (τ). The sensitivity is increased slightly by 1.69% for tracking results on the 7-camera detection volumes which indicates that similar tracking results can

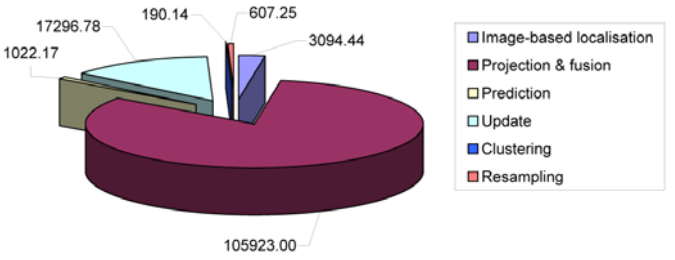


Fig. 13 Average per frame computation cost in milliseconds for image-based localization, projection and fusion on top view and track-before-detect tracking on color video from 6 cameras having a resolution of 960×544 pixels.

be obtained in cases where the top mounted cameras with fish eye lens are not available, as in most multi-camera setups. The shift in precision and sensitivity for different cameras is due to the difference in the reprojection error on the image plane.

The computational cost for creating the detection volume and for multi-target track-before-detect particle filter tracking is shown in Fig. 13. The cost refers to a C/C++ implementation for image-based localization and a Matlab implementation of the projection and fusion and MT-TBD-PF. The cost is computed per frame in milliseconds using an input video from 6 cameras having a resolution of 960×544 and detection volume whose base resolution is 492×288 on a Intel Core 2 Quad CPU having speed of 2.39 GHz and 3.25GB RAM. It can be seen that the generation of the detection volume takes approximately 85% of the processing time, whereas MT-TBD-PF takes the remaining 15% of the time (i.e., approximately 19 seconds per frame with 20 targets and 3000 particles per target). The major bottleneck in MT-TBD-PF is the update step, which takes approximately 17 seconds per frame. This computation time could be greatly reduced by using an efficient C/C++ implementation on Graphical Processing Units (GPUs) [13].

5 Conclusions

An important step in multi-camera tracking is the fusion of information from multiple views. This fusion can be performed either prior to tracking or after the initial detection and tracking step has been performed on the individual views. This chapter discussed these two categories of multi-camera tracking, which are also known as fuse-first and track-first strategies, respectively. In fuse-first strategies the target detection step that is usually required prior to tracking can be avoided by employing simultaneous detection and tracking approaches that do not required explicit thresholding of the data. In this context, a simultaneous detection and tracking approach known as multi-camera track-before-detect is discussed. This approach not only eliminates the detection step after data fusion but also implicitly helps in reducing false positives due to noise in the detection volume. The algorithm can be

implemented using sampling importance resampling particle filtering. Multi-camera information fusion for the generation of the detection volume is a computationally expensive step compared to the simultaneous detection and tracking step that incurs a relatively low cost while increasing the overall robustness against noisy measurements.

References

1. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Computing Surveys 38(4), 1–45 (2006)
2. Anjum, N., Cavallaro, A.: Trajectory association and fusion across partially overlapping cameras. In: Proc. of IEEE Int. Conf. on Advanced Video and Signal Based Surveillance, Genova, IT (September 2009)
3. Du, W., Piater, J.H.: 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)
4. Stauffer, C., Tieu, K.: Automated multi-camera planar tracking correspondence modeling. In: Proc. of IEEE Int. Conf. on Computer Vision and Pattern Recognition, Madison, WI, USA (July 2003)
5. Kang, J., Cohen, I., Medioni, G.: Continuous tracking within and across camera streams. In: Proc. of IEEE Int. Conf. on Computer Vision and Pattern Recognition, Madison, WI, USA (June 2003)
6. Morariu, V.I., Camps, O.I.: Modeling correspondences for multi-camera tracking using nonlinear manifold learning and target dynamics. In: Proc. of IEEE Int. Conf. on Computer Vision and Pattern Recognition, NY, USA (June 2006)
7. Pham, N.T., Huang, W.: Tracking multiple speakers using CPHD filter. In: Proc. of ACM Int. Conf. on Multimedia, Bavaria, DE (September 2007)
8. Black, J., Ellis, T., Rosin, P.: Multi view image surveillance and tracking. In: IEEE Int. Workshop on Motion and Video Computing, Orlando, FL, USA (December 2002)
9. Qu, W., Schonfeld, D., Mohamed, M.: Distributed Bayesian multiple-target tracking in crowded environments using multiple collaborative cameras. EURASIP Journal on Applied Signal Processing (1) (March 2007)
10. Cai, Q., Aggarwal, J.K.: Tracking human motion in structured environments using a distributed-camera system. IEEE Trans. on Pattern Analysis and Machine Intelligence 21(11), 1241–1247 (1999)
11. Khan, S., Shah, M.: Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. IEEE Trans. on Pattern Analysis and Machine Intelligence 25(10), 1355–1360 (2003)
12. Quaritsch, M., Kreuzthaler, M., Rinner, B., Bischof, H., Strobl, B.: Autonomous multicamera tracking on embedded smart cameras. EURASIP Journal on Embedded Systems (October 2007)
13. Khan, S.M., Shah, M.: Tracking multiple occluding people by localizing on multiple scene planes. IEEE Trans. on Pattern Analysis and Machine Intelligence 31(3), 505–519 (2009)
14. Fleuret, F., Berclaz, J., Lengagne, R., Fua, P.: Multicamera people tracking with a probabilistic occupancy map. IEEE Trans. on Pattern Analysis and Machine Intelligence 30(2), 267–282 (2008)
15. Eshel, R., Moses, Y.: Homography based multiple camera detection and tracking of people in a dense crowd. In: Proc. of IEEE Int. Conf. on Computer Vision and Pattern Recognition, Anchorage, AK, USA (June 2008)

16. 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)
17. Delannay, D., Danhier, N., De Vleeschouwer, C.: Detection and recognition of sports (wo)man from multiple views. In: Proc. of ACM/IEEE Int. Conf. on Distributed Smart Cameras, Como, IT (August 2009)
18. 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 (2007)
19. Hadzagic, M., Michalska, H., Lefebvre, E.: Track-before detect methods in tracking low-observable targets: A survey. On-line Magzine: Sensors and Transducers, Special Issue on Multisensor Data and Information Processing 7(2) (August 2005)
20. Taj, M., Cavallaro, A.: Multi-camera track-before-detect. In: Proc. of ACM/IEEE Int. Conf. on Distributed Smart Cameras, Como, IT (August 2009)
21. Ristic, B., Arulampalam, S., Gordon, N.: Beyond the Kalman Filter: Particle Filters for Tracking Applications. Artech House, London (2004)
22. Bruno, M.G.S., Moura, J.M.F.: Multiframe detector/tracker: optimal performance. IEEE Trans. on Aerospace and Electronic Systems 37(3), 925–945 (2001)
23. Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking. IEEE Trans. on Signal Processing 50(2), 174–188 (2002)
24. Salmond, D.J., Birch, H.: A particle filter for track-before-detect. In: Proc. of the American Control Conference, Arlington, VA, USA (June 2001)
25. Boers, Y., Driessen, J.N.: Multitarget particle filter track before detect application. IEE Proc.-Radar Sonar Navig. 151(6), 1271–1281 (2004)
26. Fallon, M., Godsill, S.J.: Multi target acoustic source tracking using track before detect. In: IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY, USA (October 2007)
27. Taj, M., Cavallaro, A.: Multi-view multi-object detection and tracking. In: Computer Vision: Detection, Recognition and Reconstruction, ch. 8, pp. 263–280. Springer Verlag GmbH (2010)
28. Taj, M., Cavallaro, A.: Distributed and decentralized multi-camera tracking: a survey. IEEE Signal Processing Magazine 28, 46–58 (2011)
29. Zisserman, A., Hartley, R.I.: Multiple View Geometry in Computer Vision. Cambridge University Press, U.K (2004)
30. 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)
31. Doucet, A., Godsill, S., Andrieu, C.: On sequential monte carlo sampling methods for Bayesian filtering. Statistics and Computing 10(3), 197–208 (2000)
32. Jia, Z., Balasuriya, A., Challa, S.: Vision based data fusion for autonomous vehicles target tracking using interacting multiple dynamic models. Elsevier Journal of Computer Vision and Image Understanding 109(1), 1–21 (2008)
33. Vermaak, J., Doucet, A., Perez, P.: Maintaining multimodality through mixture tracking. In: Proc. of IEEE Int. Conf. on Computer Vision, Nice, FR, vol. 2 (October 2003)
34. Comaniciu, D., Meer, P.: Distribution free decomposition of multivariate data. IEEE Trans. on Pattern Analysis and Machine Intelligence 2(1), 22–30 (1999)

Applications of Computer Vision to Vehicles: An Extreme Test

Alberto Broggi, Stefano Cattani, Paolo Medici, and Paolo Zani

1 Motivation

VisLab has been pioneering the world of autonomous driving since its early years; in 1998 VisLab organized one of the most innovative experiments for that period: a passenger car was equipped with sensing and actuation devices and was tested with autonomous steering along a 2000+ km on Italian highways [5].

VisLab then contiuned its efforts within this very promising research domain; it partnered with different companies and implemented the perception system of TerraMax, the largest entry in the DARPA Grand Challenge. In 2005 TerraMax was one of only 5 vehicles to successdully finish the race: about 220 km in autonomous mode along the Mohave desert in Nevada [4].

Two years later, in 2007, VisLab took part in the next DARPA event -the Urban Challenge- again on the TerraMax vehicle, which successfully qualified to take part in the final event, but -during the race- suffered a hardware problem which prevented it from finishing the test [9].

After these very successful experiments and implementations, VisLab decided to undertake a challenge on its own: it started a new project aimed at fielding a vehicle prototype able to move autonomously in most conditions. The vehicle's name is BRAiVE, short for BRAin-drIVE, and incorporates a large number of sensors and a unique sensing technology. BRAiVE has been used as a test bed for the development of a number of driver assistance systems based on the detction of vehicles, lane markings, pedestrians, obstacles, traffic signs, parking spots, and many other characteristics of the automotive environment.

BRAiVE was first presented at the IEEE Intelligent Vehicles Symplsium IV 2009 in Xi'an, China, and demoed to the participants. Thereafter it was also demoed as an autonomous vehicle in many occasions, such as in downtown Rome, form the

Alberto Broggi · Stefano Cattani · Paolo Medici · Paolo Zani
VisLab.it - Artificial Vision and Intelligent Systems Lab, IT
e-mail: {broggi,cattani,medici,zani}@vislab.it

R. Cipolla et al. (Eds.): Machine Learning for Computer Vision, SCI 411, pp. 215–250.
springerlink.com

Fig. 1 The path of the autonomous driving demonstration held in Rome in September 2009.

Capitoline Hill (Campidoglio) to the Colosseum, along the path shown in Fig. 1, in September 2009.

All demos were successful since they were carefully prepared, the parameters fine tuned, and the thresholds correctly set for that particular environment, illumination and scenario. VisLab, then, felt the need to challenge its sensing systems along a set of unpredictable scenarios, in order to check their performance in real world settings, without the possibility to change parameters and adapt them to the current situation.

VisLab therefore conceived a unique test: the idea was to drive for more than 10,000 km, day and night, on a set of uncontrolled scenarios, along unknown routes. The final decision was to drive from Parma, Italy, to the World Expo, that in 2010 was in Shanghai, China: 13,000 km to be driven by autonomous vehicles in roughtly 3 months.

As a challenge into the challenge, VisLab selected electric vehicles to be equipped with driverless technology. Nothing was changed in the original vehicles in order not to decrease their original performance both in terms of speed and distance autonomy. In fact, a solar panel was mounted on top of the vehicle to power all driverless technology: sensors, actuation devices, processing systems, satellite systems, radio,...

This had to be the most extreme test so far for autonomous vehicles in the history of vehicular robotics; other papers [**?**, **?**] deal with logistic aspects and describe the effort, while this paper describes the vision processing implemented on the vehicles and the main results.

2 Sensors

2.1 Design Considerations

Sensors positioning has been based on VisLab past experience with other vehicles and other projects[3, 7, 8]; however, the design of the vehicles is radically differ-

(a) (b)

Fig. 2 Front and back views of the electric vans used during VIAC, with accessible sensors.

ent from the one adopted on BRAiVE [2], the most advanced prototype built by VisLab to date. On the electric vans used during VIAC all the equipment has been kept accessible for maintenance even in remote locations and in extreme conditions: all sensors have been placed in easily reachable positions, cables and wires are easily identifiable, and PCs have been installed in a position that allows fast access from both the booth and the internal back seats. On the other hand BRAiVE was equipped with a completely different criterion in mind, as can be seen comparing Fig. 1 and Fig. 2: all sensors and cabling were hidden, all actuation devices were moved under the hood, and special care was taken to provide the car with a clean and tidy appearance. The look of the new vehicles has been taken into account too, albeit from a different perspective: given the number of media events planned during the trip, keeping all the extra equipment in plain sight attracted more attention, and made explaining the vehicles capabilities much easier.

All the electric vehicles have been equipped with the very same sensing, processing, and actuation technologies, but can be used in different configurations (e.g., as leader, followers, or standalone). The choice of setting up identical hardware and software on each vehicle meant increasing the development time, but the possibility of swapping the vehicles in operation has provided additional flexibility in case of failures, and has allowed to increase the number of driving shifts, overcoming the limited batteries autonomy.

No particularly expensive sensor has been considered in the design as well as no sensor with special needs in terms of physical installation has been included. This has been done on purpose, since the test aimed at evaluating technologies with widespread practical applications. Special emphasis has been given to computer vision since it provides a cost-effective way of sensing the environment; moreover it has another great advantage over laserscanners: cameras can be installed in a variety of positions on the vehicle (inside the cabin, on the roof, or within the headlights), while laserscanners used for mid-long term sensing need to be placed in front of it, usually behind the front bumper, where they are easily hit by rocks, debris, and other objects.

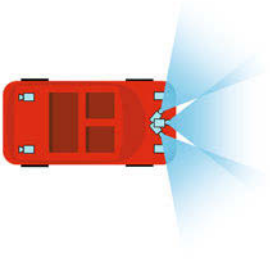
As shown in Fig. 2(a)(b) most of the sensors have been mounted outside the vehicle, exposed to any kind of weather conditions.

2.2 *Perception Systems*

A total of seven cameras (5 forward and 2 backward looking) and four laserscanners with different characteristics monitor all the area surrounding the vehicle, while a unit combining a radio, a GPS and an IMU mounted on top of the van provides self-localization and vehicle-to-vehicle communication capabilities. The following sections contain a brief description of each perception system.

2.2.1 Panoramic Vision

The panoramic vision system (Fig. 3) provides a 180 degrees view of the space in front of the vehicle by combining the images coming from 3 IEEE1394-A



(a)

(b)

Fig. 3 Panoramic Vision System: (a) the area monitored by the sensors, and (b) the cameras installed behind the windshield.

752 × 480 synchronized cameras, with 3.5 mm, 1/3 ” lenses. The resulting high resolution image is used to detect and track the leader vehicle even when approaching a tight curve or a steep hill.

2.2.2 Lateral and Front LIDARs

Two single beam laserscanners are mounted right on the corners of the front bumper (Fig. 4(a)(c)); they are used for close-range obstacle detection. Each laserscanner has an aperture of about 270 degrees, while the perception depth is about 30 m. Conversely, the front LIDAR unit is used to detect the presence of obstacles in front of the vehicle in the mid-far range (Fig. 4(b)(c)); its four scan planes allow robust detection even in case of strong pitching or terrain slopes up to 80 m, with an aperture of about 100 degrees.

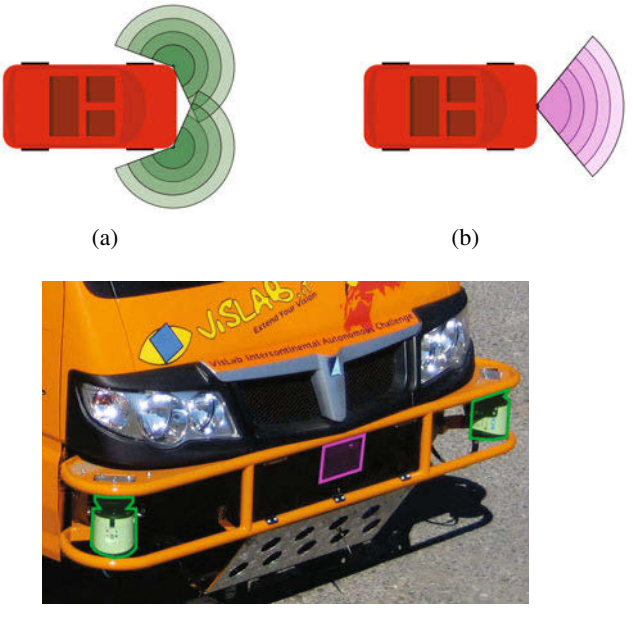


Fig. 4 Lateral and front LIDARs: (a) the area covered by the lateral sensors and (b) by the front one; (c) the laser units, mounted within the front bumper.

2.2.3 Front Stereo Vision and Off-Road LIDAR

The frontal stereo system (Fig. 5(a)(c)) exploits two 752 × 480 IEE1394-A cameras with 4 mm, 1/3” lenses and a baseline of 80 cm to locate obstacles, and lane markings in front of the vehicle up to 40 m. While both this system and the front LIDAR

(a) (b)

(c)

Fig. 5 Front stereo vision and off-road LIDAR: (a) the area covered by the front cameras and (b) by the off-road laserscanner; (c) the units mounted under the solar panel.

(a) (b)

(c)

Fig. 6 Back stereo vision and GPS/IMU unit.

cover the same area, they perform the detection task using technologies with different modes of failure (e.g., low light conditions vs dust), thus increasing the overall reliability. The mono beam laserscanner is pitched down so that the beam hits the ground in front of the vehicle (Fig. 5(b)(c)) in order to provide information about the presence of nearby ditches, bumps or rocks, especially when driving off-road.

2.2.4 Back Stereo Vision and GPS/IMU Unit

The rear stereo system (Fig. 6(a)(c)) is used to locate obstacles nearby the vehicle during backup manouvres. The GPS and inertial unit installed on top (Fig. 6(b)(c)) is used for vehicle localization; when running in leader-follwer mode the integrated radio allows inter-vehicle communication.

3 Processing Systems

The sensors layout described in the previous Section provides extended sensing capabilities. Some of the algorithm that exploit these potentials are directly taken from VisLab's BRAiVE [2] experience, such as lane detection and stereo obstacle detection. Other algorithms were developed specifically for the VIAC expedition, to meet the very peculiar requirements of such an ambitious challenge.

The sensing algorithms running on the vehicles are:

- **Lane Detection:** lane detection is a basic, yet fundamental component of autonomous navigation; our algorithm requires just one camera and it is able to detect both solid and dashed markings;
- **Stereo Obstacle Detection:** based on stereo SGM disparity engine, it provides dense depth maps, in real time, and a reliable obstacle detection, taking into account also the physical characteristics of the ego-vehicle
- **Laser Obstacle Detection:** putting together data coming from single and multi beam laserscanners, this algorithm is able to perform a robust obstacle detection and terrain removal;
- **Vehicle Detection:** when running in Leader Follower mode, the following vehicle needs to detect the position of the leader vehicle; this is made coupling laserscanner processing, for candidate selection, and vision processing, for candidate validation and leader identification.

3.1 Lane Detection

The overall system architecture is presented in Fig. 7: first an Inverse Perspective Mapping (IPM) transformation [19, 1] is applied to the frame grabbed by the camera, then a low-level filtering is performed to highlight the Dark-Light-Dark [23] (DLD) patterns of the image; the resulting points are grouped together, and clusters are finally approximated by continuous piecewise-linear functions. Once the low-level processing is over, the resulting segment lists are compared to existing lane markings in a tracking stage (Fig. 7), which produces a set of candidate lane markings; moreover, non-tracked segments are also analyzed to extract additional candidates. An expansion step is then performed, in order to join in any pertinent non-connected component still present. Finally each candidate is assigned a score

which is tested against an acceptance threshold to produce the end result. The lane markings detection is carried out for solid and dashed markings, using slightly different algorithms, and the whole procedure is performed two times, one for single and one for double lines.

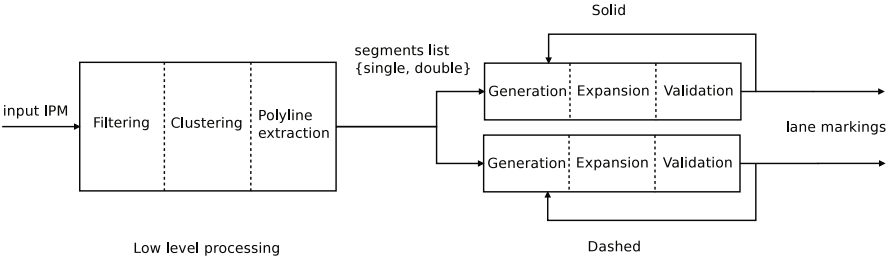


Fig. 7 System architecture.

3.1.1 Low Level Processing

The low-level processing stage derives from the one used during the 2007 DARPA Urban Challenge, described in [6]. First DLD and DLDLD transitions (with the former corresponding to single lane marking, and the latter to double ones) are extracted from the IPM and stored in separate buffers; this operation is fast since the filtering kernel is of constant size (5 and 11 pixels respectively).

After pattern extraction, a binarization step is performed: this process employs a variable threshold proportional to the average luminance of the region, over a window of size 32×1 pixels; this helps to reduce the effects of shadows cast by vehicles and roadside elements, like buildings, trees, and guard-rails. The resulting pixels are then grouped together by a clustering algorithm which uses the expansion mask illustrated in Fig. 8, with the processing taking place starting from the bottom of the image. The various groups of points are then approximated using

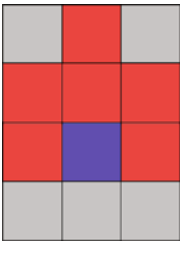


Fig. 8 Clustering algorithm expansion mask: in blue, the reference pixel; in red, the candidates for expansion. The topmost candidate helps to reduce the likelihood of cluster fragmentation due to non-continuous groups of points (e.g. because of dirty or faded lane markings).

piecewise-linear functions, so that each node on the polyline corresponds to an element of the cluster, and the maximum distance between any pixel within the label and the closest segment is below a given threshold. While different solutions to this problem exist (like the one described in [13]), the one adopted in this paper has proven to be good enough to deal with the data produced by the preprocessing stage.

3.1.2 Lane Marking Extraction

Once the set of polylines has been extracted from the image the actual lane markings extraction can take place. This process consists of three fundamental stages:

- tracking of markings detected in previous frames
- generation of new candidates
- expansion of the whole set of candidates (new and tracked)

that are carried out in that order both for solid and dashed markings, and are detailed in the following.

3.1.3 Tracking and Generation

Candidate polylines are matched against existing lane markings, and only those resulting close enough can be considered a valid correspondence. The fundamental issue to solve when performing this task is to define a distance function: a number of well-known approaches to this problem exist, like the Hausdorff [14], Frechét [11] and minimum Euclidean distances [24], but while they exhibit some interesting mathematical properties, sometimes they can produce counter-intuitive results; instead, area-based algorithms [21] seem to be more appropriate in this kind of applications: building on this idea,

the distance between two polylines a and b becomes

$$d(a,b) = \max\left(\frac{2 \times area(a,b)}{length(a)_{[V_0,V_1]} + length(b)_{[V_3,V_2]}}, d_{min}(a,b)\right) \tag{1}$$

where $length(p)_{[V_m,V_n]}$ denotes the length of the polyline portion delimited by the points V_m and V_n, $area(a,b)$ is the area between the polylines (marked in light blue in Fig. 9), and $d_{min}(a,b)$ is the minimum distance between a and b.

When performing tracking, each solid lane marking $ts_i \in \{ts_0 \dots ts_k\}$ identified at time $T-1$ is used to compute the score

$$score(ts_i, cs_j) = \frac{d(ts_i, cs_j)}{length(cs_j)}, j \in \{0 \dots h-1\} \tag{2}$$

matching it against the candidates $\{cs_0 \dots cs_{h-1}\}$ generated by the low-level processing stage at time T; the candidate obtaining the lowest value is selected, and used in the following expansion stage.

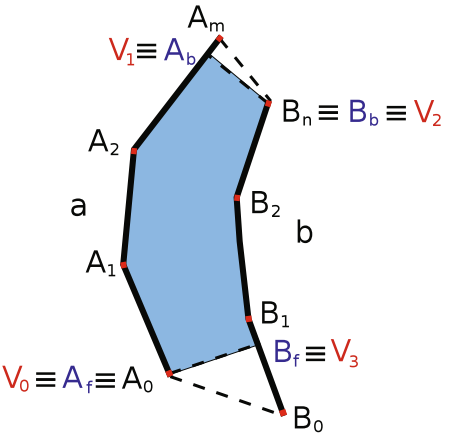


Fig. 9 Distance between polylines. In light blue, the overlapping area; in red, boundary vertexes; in blue, the projections of one polyline ends onto the other.

Dashed lane marking tracking uses a different comparison criterion, since the length of candidate dashes $\{cd_0 \dots cd_{k-1}\}$ is usually too small to be reliable:

$$score(td_i, cd_j) = d(td_i, cd_j)^2 + x_j^2, j \in \{0 \dots k-1\} \tag{3}$$

with x_j being the x component of the first vertex of cd_j. This heuristic allows to obtain as a winning candidate a dash that is near to the lane marking to track while also being as close as possible to the camera, that is, in the IPM region that is more likely to produce accurate results.

After the first dash has been determined, it is used as a starting point to join in any other dash close to the originating lane marking (that is, td_i), iteratively building a new polyline cp. The criterion adopted to perform this task aims at isolating candidates being close both to cp and to td_i, sorting dashes according to the following comparison rule:

$$\min(cd_h, cd_k) = \begin{cases} cd_h \text{ if } d(cd_h, td_i) < th \text{ and } d(cd_k, td_i) > th \\ cd_k \text{ if } d(cd_h, td_i) > th \text{ and } d(cd_k, td_i) < th \\ \underset{cd \in \{cd_h, cd_k\}}{\arg\min} \; (d(cd, cp) + d(cd, td_i)) \text{ otherwise} \end{cases} \tag{4}$$

Each time a dash is added to cp the remaining ones are sorted again using Eq. 4, until no close dashes are left. To further improve the robustness of this step, dashes are joined only if they satisfy the constraints illustrated in Fig. 10, and further explained in 3.1.4.

3.1.4 Expansion and Validation

Tracked and non-tracked polylines are iteratively analyzed to determine whether any other compatible candidate exists; if it is found, its points are merged in, and the search continues using the resulting polyline as the new reference. For both solid and dashed lane markings, a common condition for inclusion is that the first vertex of the candidate polyline must be close to the last vertex of the reference one, as it is illustrated in Fig. 10; moreover, the orientation of the end segments must be similar (that is, the angle φ in Fig. 10 must be small), and the connection angle (θ in Fig. 10) must also be small. Dashed markings bear the additional constraint that dash lengths and pauses between dashes must have a similar length.

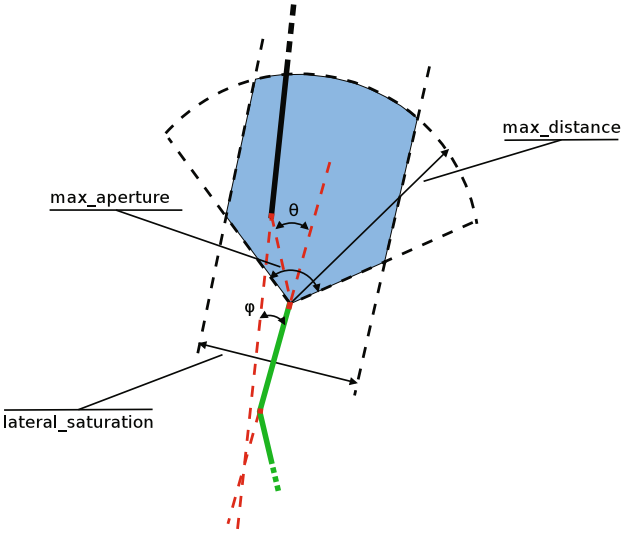


Fig. 10 Compatibility test. In green, reference marking, in black, candidate polyline to join. The candidate is considered for inclusion only if its first vertex falls inside the blue area (determined by the parameters *max_distance*, *max_aperture* and *lateral_saturation*), and the angles φ and θ are small enough.

When the search is over, a score is assigned to the resulting polyline p, to determine whether it should be accepted as a valid lane marking or not. The value is computed as

$$score(p) = \frac{length(p)^2}{d((0,0),p)^2} \tag{5}$$

and if p has been successfully tracked, the old score is added to the current. Using this approach means that lines starting far away from the vehicle are considered valid only if they are consistently detected over a high number of frames, while close, long lines are more easily accepted.

Tracked lane markings are not discarded immediately in case of a missed detection; instead, they are kept as "ghosts" for up to 5 frames.[1]

3.1.5 Results

The algorithm proposed in this paper has been tested on image streams captured in both highway and urban scenarios, in a variety of conditions. Fig. 11 contains some samples from different image sequences, along with lane detections results.

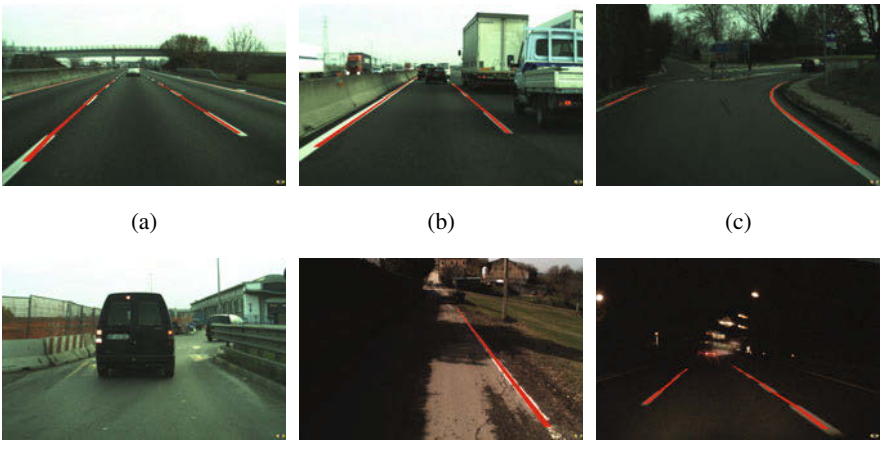


Fig. 11 Some sample outputs in different situations: (a) Highway scenario; (b) Heavy traffic; (c) A downhill intersection; (d) Construction area, with no false detections; (e) Uphill motorway with shadows; (f) Nighttime highway under heavy rain.

In ideal working conditions outputs are correct (Fig. 11(a)), even in case of heavy traffic (Fig. 11(b)), and tracking is very stable, with lines being continuously identified for up to 600 frames. Since no predefined model of the road is enforced atypical geometries can be handled as well, like in Fig. 11(c), where lane markings leading to an intersection at the end of a downhill road are detected correctly. The adopted score criterion shows its effectiveness in Fig. 11(d): no false detection is introduced, despite the number of objects present in the scene, including a vehicle right in front of the camera, the striped barriers on the left and the guard-rail on the right. Fig. 11(e) and Fig. 11(f) contain the results obtained in two challengig situations, namely on a motorway with a lot of shadows, and under heavy rain at night: in both cases the algorithm correctly detects the road markings.

[1] The value has been determined empirically.

3.2 Stereo Obstacle Detection

Processing happens following the steps presented in Fig. 12.

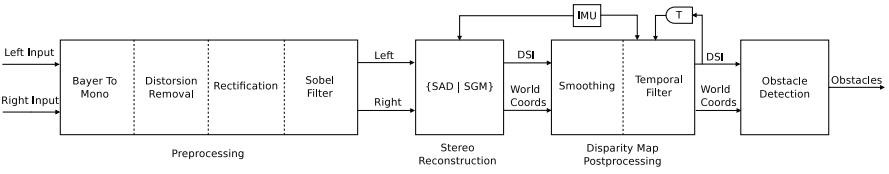


Fig. 12 System architecture.

- *low-level processing* — Bayer-patterned images acquired by the cameras are converted to gray-scale, then lens distortion is corrected, and the stereo pair is rectified. A vertical Sobel filtering is used to improve the subsequent matching phase;
- *disparity map generation* — stereo reconstruction is carried out; both correlation-based and Semi-Global Matching (SGM) approaches have been tested ;
- *disparity map post-processing* — two filters are applied to further enhance the map quality
- *obstacle detection* — actual obstacle detection takes place

3.2.1 Stereo Reconstruction

Given the tight development schedule, Disparity Space Image (DSI) generation was implemented exploiting the already available window-based SAD correlation technique described in [10], with satisfactory results; nevertheless, in the months following VIAC an efficient implementation of the SGM algorithm has been devised, which performs even better.

In order to generate a Disparity Space Image D the Semi-Global Matching algorithm performs an energy minimization step. The energy function $E(D)$ that has to be globally minimized consists of two terms: the pixel-wise matching cost $E_{data}(D)$ and the smoothness constraint $E_{smooth}(D)$:

$$E(D) = E_{data}(D) + E_{smooth}(D) \tag{6}$$

The $E_{data}(D)$ term is the sum of all pixel matching costs C for the disparities of D:

$$E_{data}(D) = \sum_{\mathbf{p}} C(\mathbf{p}, D_{\mathbf{p}}) \tag{7}$$

Instead of using mutual information as the pixel-wise matching function, as it is done in [16], the Hamming distance of the Census transform of a 5×5 window cropped around $\mathbf{p}$ has been exploited, since it provides similar results [17, 12] while reducing the overall computational burden.

The E_{smooth} term adds a small penalty P_1 to all pixels $\mathbf{q}$ in the neighborhood $N\mathbf{p}$ of $\mathbf{p}$, for which the disparity varies from $\mathbf{p}$ by one, and a higher penalty P_2 if the difference is greater:

$$E_{smooth} = \sum_{\mathbf{q} \in N\mathbf{p}} P_1 \mathbf{T}[|D_\mathbf{p} - D_\mathbf{q}| = 1] + \sum_{\mathbf{q} \in N\mathbf{p}} P_2 \mathbf{T}[|D_\mathbf{p} - D_\mathbf{q}| > 1]$$

with

$$\mathbf{T}[x] = \begin{cases} 1 \text{ if } x \text{ is true} \\ 0 \text{ otherwise} \end{cases}$$

The global minimization of $E(D)$ is a NP complete problem, that SGM approximates by computing the values of $E(D)$ along 1D paths from 8 directions towards each pixel using dynamic programming. The costs $L'_\mathbf{r}$ of each path $\mathbf{r}$ are aggregated as described in Eq. 9 for each pixel $\mathbf{p}$ and disparity d:

$$\begin{aligned} L'_\mathbf{r}(\mathbf{p},d) = C(\mathbf{p},d) + \min(&L'_\mathbf{r}(\mathbf{p}-\mathbf{r},d), \\ &L'_\mathbf{r}(\mathbf{p}-\mathbf{r},d-1)+P_1, L'_\mathbf{r}(\mathbf{p}-\mathbf{r},d+1)+P_1, \\ &\min_i L'_\mathbf{r}(\mathbf{p}-\mathbf{r},i)+P_2) \end{aligned} \tag{8}$$

The final disparity value for each pixel is then determined by a winner-takes-all strategy applied to the values of $L'_\mathbf{r}$. To further improve the results sub-pixel interpolation is performed as well as a median filter and a left-right consistency check.

In order to fully exploit the parallel processing capabilities of modern multi-core CPUs and reach real-time frame rates, a multi-threaded, SIMD processing scheme has been devised. The most time-consuming step of the SGM algorithm is path accumulation, since it must be performed for each pixel, disparity, and path: to speed up the processing, for each path direction the pixels are split into several independent slices that are processed in parallel; moreover only the accumulated value is saved into memory, while temporary data needed for incremental processing is kept into the CPU registers[2]. Finally, when computing the results of Eq. 9 the Intel® SSE instruction set is also used, outputting 16 disparity values at a time.

3.2.2 DSI Post-processing

The generated disparity map is post-processed to fix possible spurious values; this is especially useful when using correlation-based stereo, where local minima are more likely to introduce noise.

A first filter analyzes a 3×3 window around each DSI element, checking that its disparity value is close to a sufficient number of neighbors, and marks it as invalid

[2] Disparity search ranges of up to 128 are supported; over this threshold not enough XMM CPU registers are available.

in case this condition is not satisfied. After this step has been performed, invalid pixels whose neighborhood has a variance lower than a fixed threshold are assigned the average value of the surrounding elements.

The second filter uses IMU information to compute the vehicle trajectory between the previous and current frame acquisition, and checks the corresponding disparity maps for consistency. At time T each point $p_i(u,v,d)$ of the depth map D_T is projected into the corresponding world point $p_w(x_w,y_w,z_w)$ [15], which is in turn projected back into DSI coordinates using virtual cameras corresponding to the position and orientation of the stereo rig at time $T-1$ with respect to the current reference system; this process generates the depth map D^T_{T-1}, which can be directly compared to D_{T-1}, computed at $T-1$. Each pixel of D^T_{T-1} is analyzed, and considered valid only if at least one of the pixels of D_{T-1} within a 3×3 window has a disparity value close enough.

3.2.3 Obstacle Detection

Since no assumptions could be made on the road infrastructure quality and the kind of traffic to expect, the obstacle detection algorithm design followed the approach first described in [20]. This technique defines a criterion to cluster points into obstacles based on their layout in space and on the physical characteristics of the ego-vehicle, namely its height H_{max}, the minimum relevant obstacle height H_{min} and the maximum traversable slope θ_{max}. Given two points p_{w1} and p_{w2} these constraints are used to define a truncated cone in space, with the vertex corresponding to p_{w1}, as depicted in Fig. 13: if p_{w2} falls within the cone it is labeled as an obstacle, and p_{w1} and p_{w2} are said to be *compatible*.

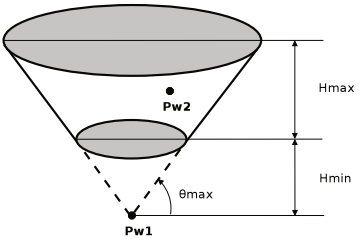


Fig. 13 (a) Compatibility criterion used during the clustering phase. Point p_{w2} is considered an obstacle since its position relative to p_{w1} leads to a path which is not traversable by the ego-vehicle.

To reduce the number of comparisons to perform, compatibility checks are carried out in image coordinates, rather than in the world: this is done by projecting the truncated cone back onto the image using camera calibration information, thus (approximately) obtaining a trapezium, and checking whether the constraint is valid for the disparity points contained within it. Iterating on the disparity space image from bottom to top, and from left to right, it is possible to correctly cluster the whole data

set. During the clustering phase each region is assigned a unique label; to ensure that a single obstacle does not get split into multiple adjacent regions the following strategy is adopted: let l_{w1} and l_{w2} be the labels of points p_{w1} and p_{w2} respectively, then

- if $l_{w1} \neq undefined$ and $l_{w2} = undefined$, $l_{w2} \leftarrow l_{w1}$
- else if $l_{w1} = undefined$ and $l_{w2} \neq undefined$, $l_{w1} \leftarrow l_{w2}$
- else if $l_{w1} \neq l_{w2}$ all the points with label l_{w2}, plus p_{w2}, are relabeled as l_{w1}

The final result is saved into two dual representations:

- an image, having the same resolution of the depth map, where each pixel value corresponds to a label ID;
- a vector of regions, each containing the list of its points.

Even using DSI coordinates the computational load associated with the clustering phase is considerable, especially when using the dense depth map produced by the SGM matching algorithm. In order to fully exploit the parallel processing capabilities of the target hardware platform, a multi-resolution, multi-threaded analysis scheme was devised.

The original DSI, along with its associated 3D world points vector, is split into N images, each containing only points corresponding to a predefined range of distances on the X axis, as illustrated in Fig. 14.

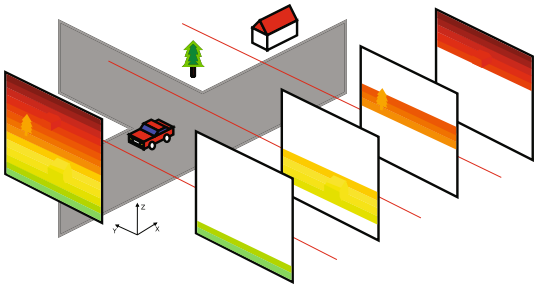


Fig. 14 DSI partitioning. Separate depth maps are created, each corresponding to a slice of world points, then the maps are analyzed in parallel on different CPU cores.

The depth maps are then sub-sampled in order to further reduce the number of comparisons to perform: ideally constant spatial resolution can be achieved [22], but in practice it is more convenient to fix a single resolution for each stripe;the partitioning has been experimentally chosen as follows:

- 0 – 5 m $\rightarrow$ 4$\times$ sub-sampling
- 5 – 15 m $\rightarrow$ 2$\times$ sub-sampling
- 15 – 30 m $\rightarrow$ full resolution
- 30 m – ∞ $\rightarrow$ full resolution

After this step each slice is processed independently in parallel, leading to N sets of labels; the sets are then copied back on a single, full-resolution map, upscaling the sub-sampled clusters, while checking that the extrapolated points are similar enough to the original ones, in order to avoid introducing blocky artifacts near depth discontinuities.

A final pass is still needed to ensure that all labels are properly merged, since an obstacle spanning across a slice boundary is still split in two distinct labels, as shown in Fig. 15. Each point in each cluster is checked to determine whether the value within the labels image corresponds to the one saved within the regions vector, and in case they are different, the two points sets are merged together and the labels image is updated accordingly.

(a) Input DSI. (b) Labels before merging. (c) Merged labels.

Fig. 15 Obstacles labeling on a steep road when driving uphill. First points are clustered according to their relative pose, in parallel, on multiple CPU cores, then the labels are merged and filtered to produce the final obstacles list. Note that despite the slope the vehicle is facing, no false detection is triggered even without performing any explicit modeling of the ground.

3.3 Laser Obstacle Detection

Laser Obstacle Detection algorithm (LOD) exploit data coming from Lateral and Front LIDARs (see Sec. Sensors) to perform obstacle detection and tracking.

The Laser Obstacle Detection processing follows these steps, shown in Fig. 16:

1. mapping and data *fusion*;
2. clustering;
3. clusters analysis and characterization;
4. clusters tracking;

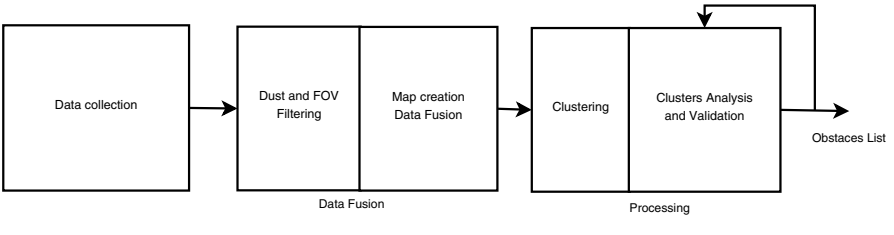


Fig. 16 System architecture.

3.3.1 Data Fusion

Lateral and Front LIDARs system are made of single and multi beam laserscanners, with overlapping fields of views (FOVs). Hence, data fusion is performed with two main aims:

- *redundancy*: inside the multi beam laserscanner's FOV there is FOVs overlapping with single beam laserscanners. Here echoes coming from single beam laserscanners are considered only if they falls in locations whereat least one multi beam laserscanner's echo is already presents. This because the single beam laserscanners are intrinsically less robust than multi beam. So, they are used to create redundant information to reinforce the multi beam information;
- *complementarity*: outside the multi beam laserscanner's FOV there is no FOVs overlapping, here only single beam laserscanner can provide information. In these regions we completely rely on the single beam laserscanners to extend the Front LIDAR; here no terrain removal is possible;

3.3.2 Mapping

Data fusion happens when each echos is projected into a bird-eye-view map of the space in front the vehicle, made of $10cm \cdot 10cm$ *cells*, like the one shown in Fig. 17.

Actually this map is stored in memory as an $238 \cdot 400$ pixels gray scale image. As shown if Fig. 17, there is a direct correspondence between pixels in row/column coordinates and cells positions in XY coordinates. So, given an echo, it will be

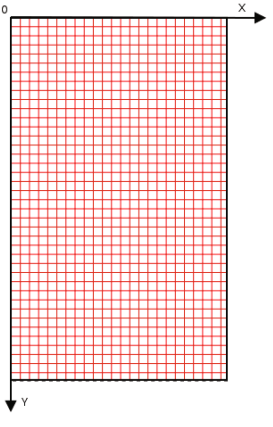


Fig. 17 Grid where LIDARs' beams are projected. Note how the reference system XY is oriented as columns/rows in a typical image buffer.

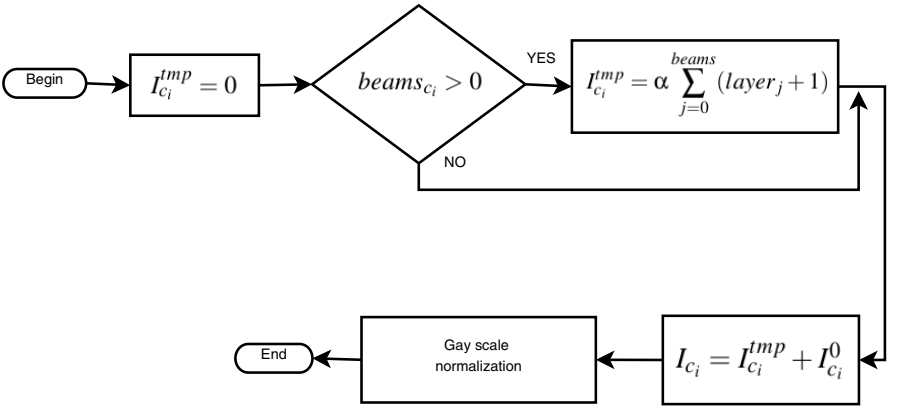


Fig. 18 Map's grey scale values creation.

assigned to the corresponding cell, increasing the cell's value: the higher the gray value, the more echoes that have fell into that cell.

Once the grey values of each cell has been computed, on the basis of the corresponding echos falling into them, values are also *shared* with each cell's neighbor to increase robustness, following this rule:

- 80% of the cell value is passed to neighbors within 1 pixel radius;
- 70% of the cell value is passed to neighbors within 4 pixel radius;
- 50% of the cell value is passed to neighbors within 9 pixel radius;
- 40% of the cell value is passed to neighbors within 13 pixel radius;

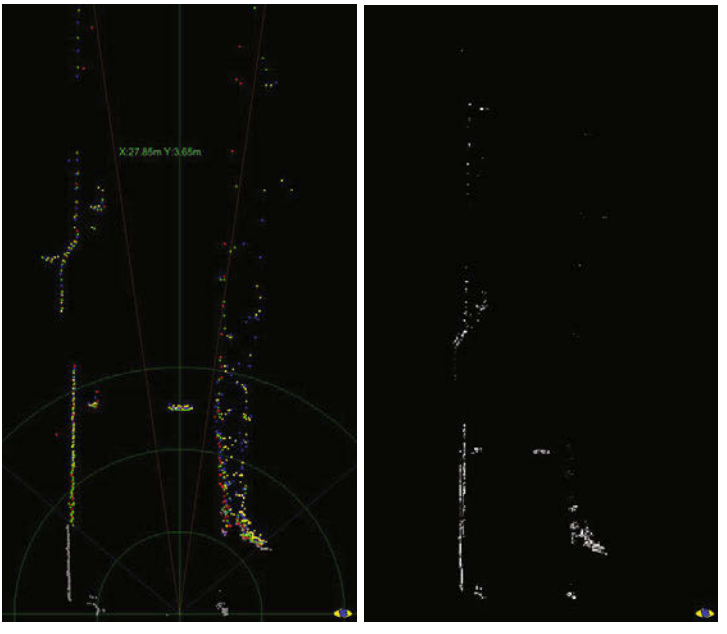
In Fig. 19 an example of map creation is shown. Note that the final map has been eroded and dilated, to remove isolated pixels and fill missing pixels.

3.3.3 Clustering

The clustering phase tries to put together adjacent cells to create a list of objects/obstacles candidates. The list will be subsequently refined to remove clusters that do not meet minimum size constrains. The clustering algorithm is shown in Fig. 20.

Basically, for each not null pixel a 3×3 neighborhood area is analyzed, to find adjacent not null pixels. At the end of this first phase, to each pixel will be assigned a grey scale value that no longer represent the number of echos fell into them, but just a *label* that identifies pixels belonging its same cluster.

In a subsequent phase a 5×5 neighborhood area is analyzed, to include those pixels that were discarded during the erode and dilate operations.



(a) Echos coming fro laserscanners

(b) Grey scale map

Fig. 19 Grey scale map generation

3.3.4 Object Moments

For each objects the corresponding Moments are computed. Image Moments are a particular weighted average of the image pixels' intensities. The simple moment of $(p+q)$ order is defined as follows:

$$M_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p y^q f(x,y)\, dx\, dy \quad p,q = 0,1,2,... \tag{9}$$

On the basis of the simple moments it is possible to compute the Hu Moments, that provides useful information about object, such as centre of mass, main axes, orientation. Moreover, Hu Moments are rotation, translation and scaling *invariants*, so they result to be very useful for shape matching (see Sec. 3.3.5).

Moments are also used to perform a first object filtering:

- M_{00} correspond to the object area;
- $\{\bar{x},\ \bar{y}\} = \{\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\}$ is the center of mass

comparing area and position (with respect of the sensor) we can discard those objects that appear to be too small.

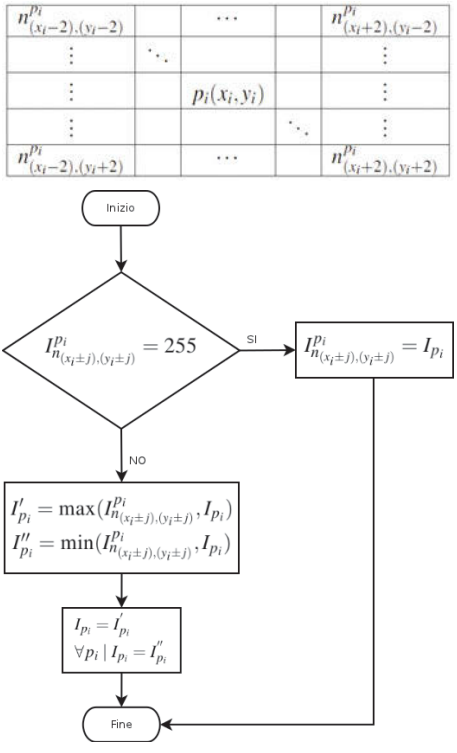


Fig. 20 Clustering neighborhood schema and clustering algorithm

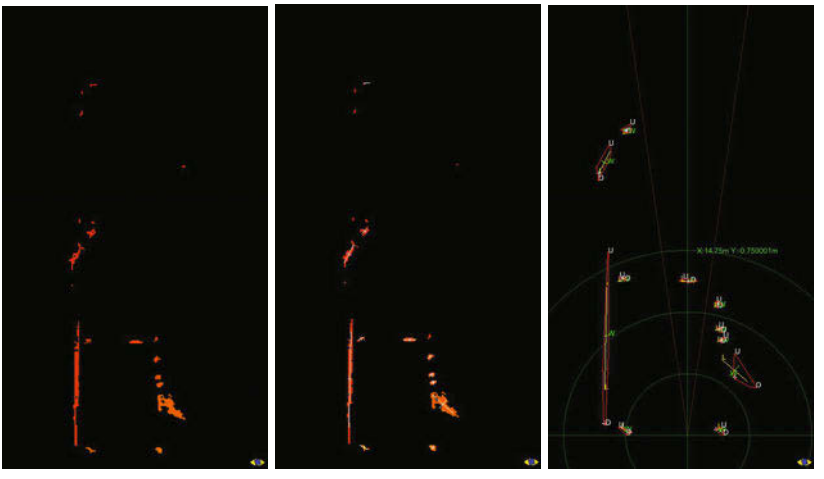
3.3.5 Tracking

There are a variety of tracking algorithm know in literture [26]. In our case the solution implemented for Laser Obstacle Detector is based on three main data structures:

- *current objects*: the list of the object detected at time current t;
- *validated objects*: the list of object that has been provided as output at time $t-1$;
- *ghost objects*: the list of validated object at time $t-2$ that have not been tracked succesfully at time $t-1$;

Each object i, validated at time $t-1$, is compared with each currently detected object j, obtaining a match value as follows:

$$f_j^i = 0.75\left(\frac{\Delta CM_{ij}}{\Delta CM^{th}}\right) + 0.25 m_{ij} \tag{10}$$



(a) Clustered cells (b) Hu Moments (c) Final output

Fig. 21 Tecnica di *labelling*

where ΔCM_{ij} center of masses interdistance, ΔCM^{th} is user define normalization factor and m_{ij} is a Hu Moments based shape match factor, computed as follows:

$$m_{ij} = \sum_{n=1\ldots7} \left| \frac{1}{m_n^i} - \frac{1}{m_n^j} \right| \begin{cases} m_n^i = \text{sign}(\phi_n^i) \cdot \log \phi_n^i \\ m_n^j = \text{sign}(\phi_n^j) \cdot \log \phi_n^j \end{cases} \tag{11}$$

If a validated object j does not have any currently detected object i that lead to a match value f_j^i above a given threahols, object j is moved into the ghost list. An object can stay into the ghost list no longer than 5 processing cycles, then it is discarded.

All the currently detected object that have not been matched with any previously validated object are directly inserted into the validated list, without any delay.

The new validated object list is delivered as the algorithm output.

3.3.6 Results

In Fig. 22 some processing results in details: (a)(b) the leader vehicle; (c)(d) a bush near the road surface; (e)(f) a bicycles; (g)(h) a gardrail.

3.4 Vehicle Detection

The vehicle detector algorithm is used by the follower vehicle to detect the position of the leader vehicle. As explained in the previous Sections, the leader vehicle is

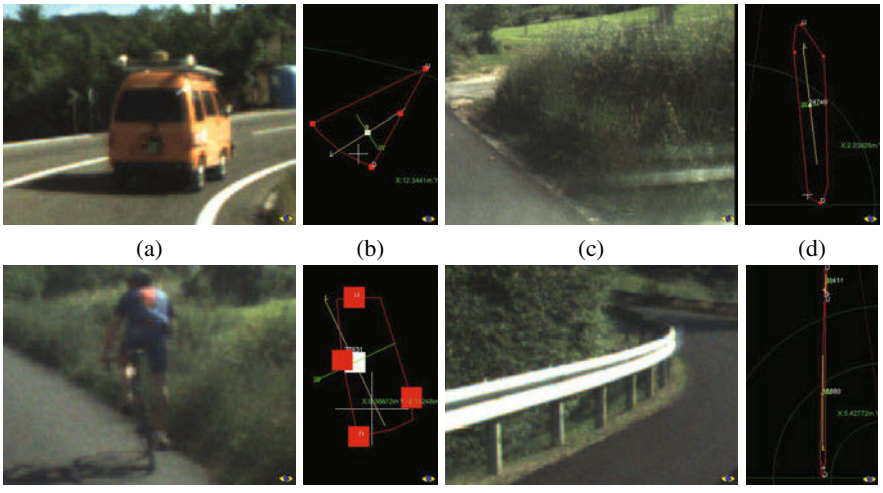


Fig. 22 Detection results details

a perfect copy of the follower, hence we know how it should appears in images and laserscanners data. We use this a-priori knowledge to improve robustness and efficiency.

Processing is based on the fusion between the two main sensors mounted onboard: a laserscanners belt placed inside the front bumper and a panoramic view system composed of three cameras placed inside the cabin in place of the rear view mirror. The main steps are shown in the flowchart of Fig 23:

- a first preprocessing phase, based on LIDAR data, finds current possible vehicle candidates;
- the panoramic image portions containing the vehicle candidates are analyzed understand if it actually contains a vehicle (e.g rear lights and leds are searched, object size, plates, symmetry, ect.);
- if a previously detected vehicle (time $t-1$) is present, the corresponding portion of the panoramic view is selected. Only this part of the image will be processed at time t, to improve efficiency; moreover, the KLT features detected at time $t-1$ around the vehicle are tracked into the new image portion;

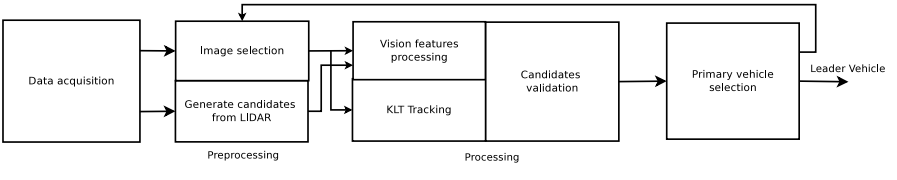


Fig. 23 System architecture.

3.4.1 Laser Preprocessing: Vehicle Back Detection

Objects list provided by Laser Obstacle Detector (see Section 3.3) is here analyzed to select a sublist of possible vehicles. Since we want to follow our leader vehicle, we are interested in detecting the rear of this vehicle, if visible, not the front nor the sides. In Fig. 24 we can see a couple of examples of how a vehicle can appear when detected by a laserscanner: it can be a kind of L or just a simple line.

Fig. 24 In general a vehivle detected by a laserscanner will appear like an L (b) or like a simple line (a)

The processing to detect the correct segment is based, again, on the object's Hu Moments: main axes are rotated of an angle α computed as follows:

$$\alpha = \Theta - \Delta\alpha,\ \Delta\alpha = \begin{cases} \frac{W_{hu}}{L_{hu}}\frac{\pi}{4} & \frac{W_{hu}}{L_{hu}}\frac{\pi}{4} < 0 \\ \frac{W_{hu}}{L_{hu}}\frac{\pi}{4} + \frac{M_{00}}{2M_{00}^{th}}\left(\frac{\pi}{4} - \frac{W_{hu}}{L_{hu}}\frac{\pi}{4}\right) & \frac{W_{hu}}{L_{hu}}\frac{\pi}{4} > 0 \end{cases} \tag{12}$$

where Θ represents the current axes orientation angle. An example of object's principal axes rotation is shown in Fig. 25(a)(b).

Once we have properly detected the orientation of the object's axes we can perform an iterative search process to precisely locate the segment position, as shown in Fig 25(c)(d). For time efficiency only one axes is searched:

1. the axes with a length closest to the target vehicle width is preferred;
2. if axes's lengths are very similar and we have a previously detected segment from time $t-1$, the axes along we search is the most perpendicular to the previously detected segment;
3. if axes's lengths are very similar and we do not have a previously detected vehicle, the nearest axes to the vehicle is preferred.

As it might already be clear from case 2 on the previous list, vehicle segments (representing the back of a vehicle) are also tracked over time with an algorithm very similar to the one described in Secion 3.3.5.

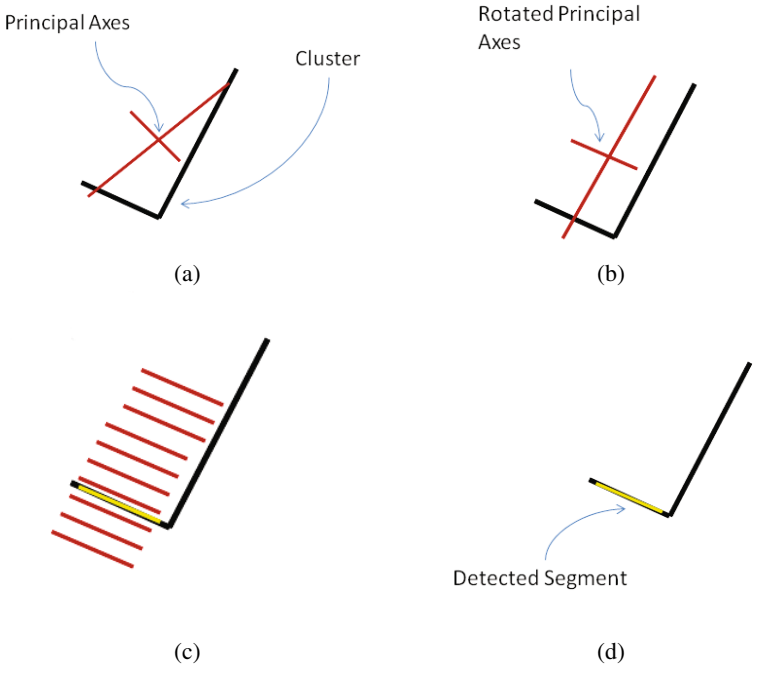


Fig. 25 Segment detection process: (a)(b) principal axes rotation; (c)(d) segment locatlization

3.4.2 Image Analysis

Once we have a list of vehicle candidates, each one represented by a 2D segment corresponding to the vehicle rear side (back), we continue with an image processing analysis of these candidates. In particular we try to understand which candidates are actually vehicles and, among the validated vehicles list, which one is the most similar to our target leader. The features alanyzed are the following:

- *symmetry*: every vehicle rear side has a high degree of symmetry; we know from the laser preprocessing its width, where is located and which is the angle of the vehicle back, hence we can set the symmetry axes right in the middle (as shown in Fig. 26(b)) and compare each pixel with the corresponding on the other side; counting how many pixels match with their corresponding we can obtain a symmetry degree value. Note how we actually compare the edges pixels, not the source image, as shown in Fig. 26(c).
- *plate*: our target vehicle has a white plate in a know position; again, on the basis of the vehicle back position obtained from the laser preprocessing, we can search for the platein a very limited area; The analysis is made by horizontal histogram of the binarized srouce image, as shown in Fig. 26(d). The presence of a plate increase the probability of the analyzed object to be validated as a vehicle;

- *rear light*: again, we know the target vehicle rear light position, color and shape; the presence of rear lights increase the probability of the analyzed object to be validated as a vehicle; the read light detection in made analyzing the binarized red image: an image made only of source's red pixels (Fig. 26(d));
- *color*: our target vehicle is mostly orange; the color analysis does not change the probability of a object of being a vehicle, but increase the probability of being the target vehicle. Hence, this processing is made only on already validated vehicles.

At the end of this analysis the objects validated as a vehilce that matches best the target vehicle characteristics (width, color, shape, etc.) is provided in output as the leader vehicle. In Fig 28 some leader detection results are shown.

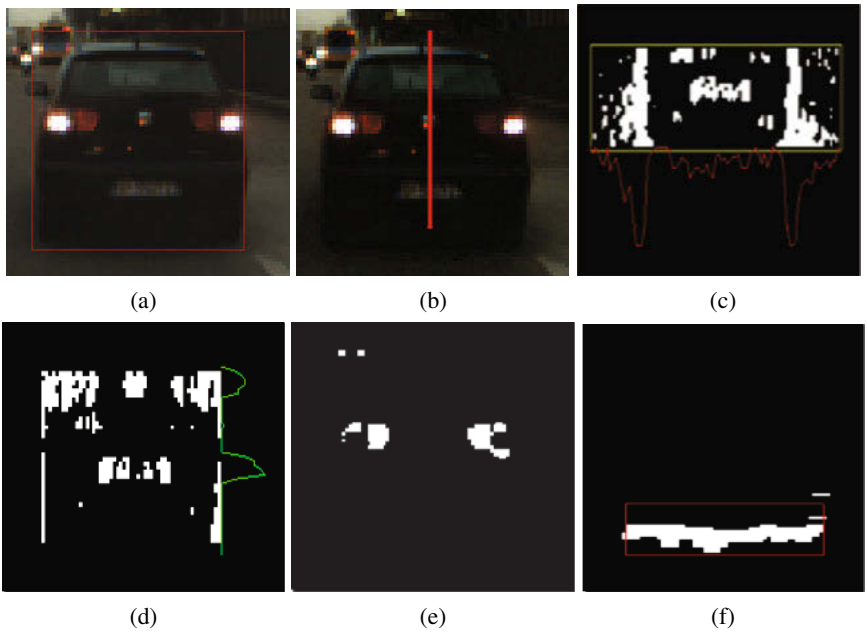


Fig. 26 Processing examples: (a) the vehicle candidate; (b) the possible symmetry axes; (c) vertical histogram of the edges image; (d) binarized red image, for rear light detection; (e) horizontal histogram, for plade detection; (f) binarized source image, thresholded for shadow detection.

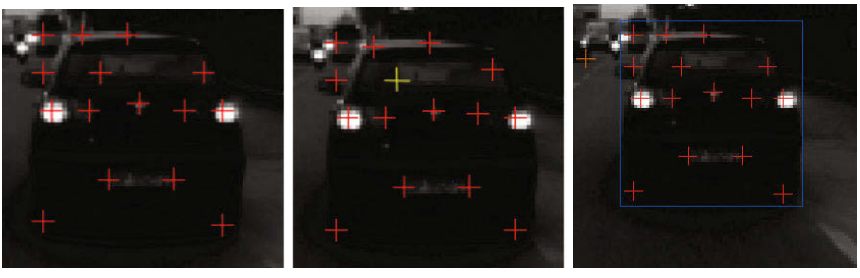
3.4.3 KLT Tracking

The leader vehicle is tracked over time by the KLT [18] [25] feature tracker:

1. when a leader vehicle is detected for the *first time*, the KLT Good Feature extractor is applied on the image protion containing the leader, to provide a list of features to be tracked in subsequent frames;
2. when a leader vehicle is detected and it seems to be the same of the previous frame (this information comes from the Laser Obstacle Detector tracker), KLT

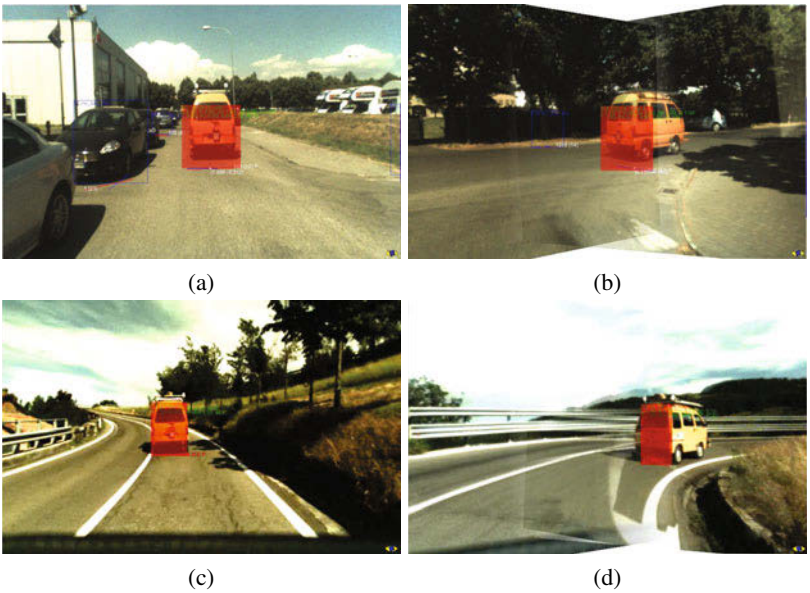
Track Features is called, to track features from the $t-1$ leader image portion to the t leader image portion;

3. during the Track Features procedure, some features might not be tracked on the new image portion; hence, after a while, the number of leader vehicle features might become too low. In this case a new set of features is computed, again calling KLT Good Feature extractor.



(a) (b) (c)

Fig. 27 KLT tracking: (a) features selected at time $t-1$; (b) the same features tracked on frame t; (c) selection of those features belonging the vehicle.



(a) (b)

(c) (d)

Fig. 28 Some detection results. Note how in (a) and (c) only the central camera is used, while in (b) and (d) a portion of the panoramic view corresponding to half central and half right camera has been used.

KLT features traking is very useful when laserscanner is not able to detect the leader vehicle. In this case the features coming from time $t-1$ are tracked, at time t, on the current image. If the tracking is sucessful, the features cloud detected on the leader vehicle at time $t-1$ are traslated on the actual vehicle position on the current image. This will give us a new hint about the vehicle position, on the basis of which we are going to select the image portion to be analyzed in the way described in Sec. 3.4.2.

After a certain number of iterations whith no valid laser detections of the leader vehicle, KLT tracking is interrupted and, consequentely, the algorithm will fail in providing the expected output.

4 Statistics

Data coming from all the sensors and the path-planning system has been recorded throughout the expedition, but using it to generate meaningful statistics for such an extensive experiment is a challenging task, due to the lack of a ground truth. To overcome this issue two strategies have been employed: in controlled environments with good GPS coverage (such as the demonstrations sites) a recorded path is used as a reference, while in totally unknown areas a manned leader vehicle has been used as a reference.

Even when the GPS signal is available, the measured error is only an approximation, given the limited accuracy provided by the unit installed on the vehicles, which does not use technologies like DGPS or GPS-RTK.

4.1 Preliminary Test

During development several preliminary tests were performed, involving a number of kilometers of autonomous driving in different weather and environmental conditions. Experimental results shown in this first part were obtained by testing the vehicles inside the 2 km long loop of the Parma University Campus (Fig. 29), and in its surroundings, both in urban and rural scenarios.

In Fig. 30(a) and 30(b) the results of WayPoint Following autonomous tests (6 laps performed inside the university campus) are presented: the mean crosstrack error reported by this experiment was of 0.13 m, and its standard deviation was of 0.15 m. The average speed on this test was 26 km/h and the maximum speed reached was 46 km/h.

Autonomous driving performance in leader follower mode, both in urban and extraurban areas are shown in Fig. 31(a) and 31(b): the mean crosstrack error is 0.17 m, and its standard deviation is 18 m. The average speed on this test was 28 km/h (maximum 50 km/h).

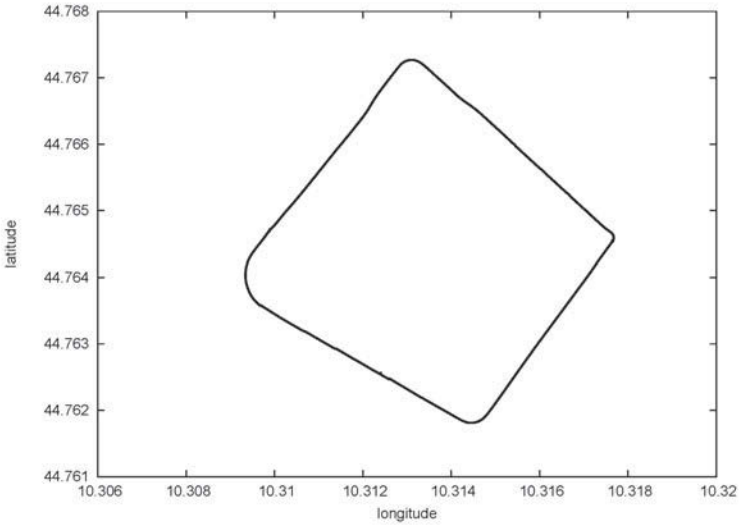
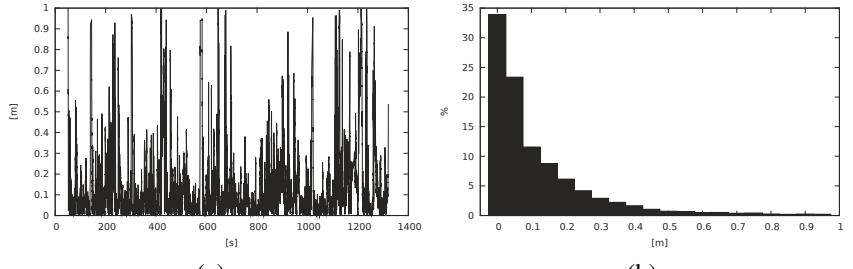


Fig. 29 The test-bed: University of Parma campus area.



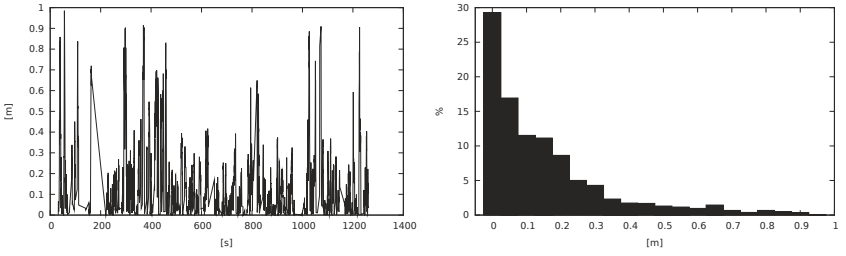
(a) (b)

Fig. 30 Plot of lateral crosstrack error *(a)* and relative histogram *(b)* during the experiment of 22 minutes of autonomous operations on campus with autonomous GPS Waypoint following.

The significant increase in the lateral crosstrack error is explained by the presence of obstacles along the road which influence the trajectory of the vehicle as it replans to avoid them, moving it away from the original WayPoint.

4.2 The Challenge Statistics

In this section the performance in autonomous mode of the 4 electrical vehicles on the whole trip is presented. Officially the expedition started on 26 of July in Parma and ended on 28 of October in Shanghai after crossing eight states and performing



(a) (b)

Fig. 31 Plot of lateral crosstrack error *(a)* and relative histogram *(b)* during the experiment of 21 minutes of autonomous operations in urban and rural environments with vehicle in autonomous Leader Vehicle following.

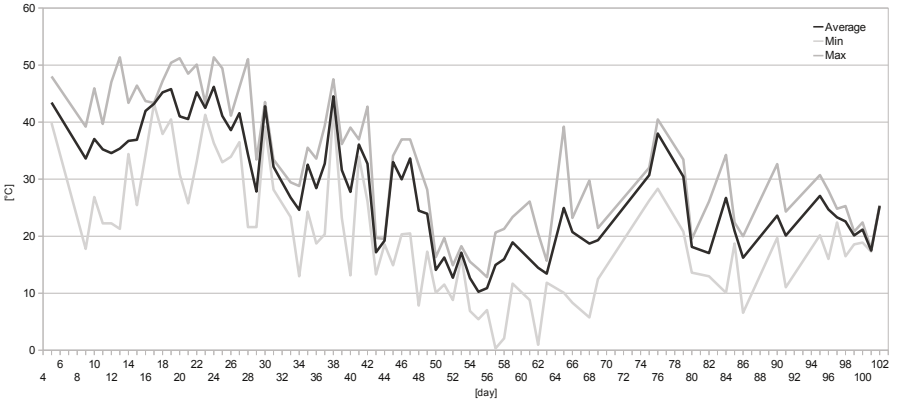


Fig. 32 Temperatures reported by inertial sensor during the whole trip.

demos in 22 different cities. The convoy was composed by 4 electric autonomous vehicles, 3 support trucks and 4 RVS.

An idea of the different environments encountered and the extreme stress conditions the components suffered can be seen from Fig. 32, which shows the temperature reported by the sensors installed on top of the vehicles.

The data collected refers to the effective 61 days of autonomous driving: altogether 214 hours were traveled divided into 191 different runs. Usually the runs ended when no battery power was left, but sometimes logistic needs mandated a stop, such as when crossing a state border. The maximum distance traveled in autonomous mode was 96.7 km. A detailed chart of travelled distance per day is shown in Fig. 33. In this graph also it is possible to see which major cities were touched, the states borders and the days when demonstrations were performed. It was not possible to travel non-stop every day for different reasons; for organizational requirements, the first week was used for demonstrations in several italian cities and to allow last-minute systems tuning. Several days during the trip were devoted to

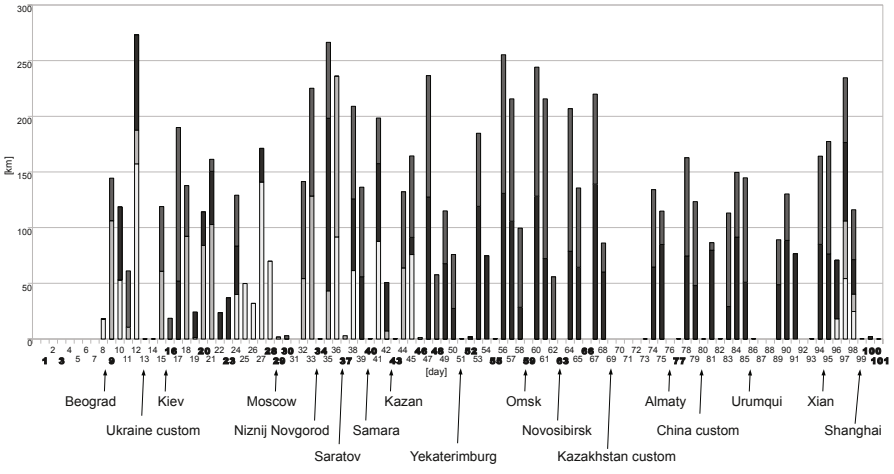


Fig. 33 Travelled distance in autonomous mode per day. Different colors are used to indicate the distance performed by each of four different vehicles. The expedition was involved in demonstrations on days marked with boldface.

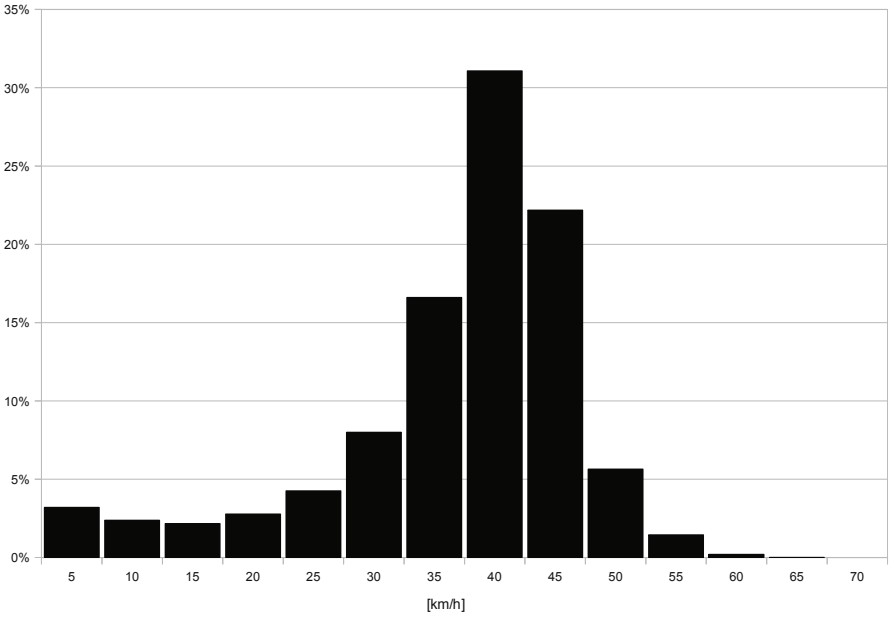


Fig. 34 Speed profile performed by the test vehicles during the whole experiment.

demonstrations of autonomous driving in the visited cities downtown. Finally other days were used to solve mechanical problems, or pass through customs. For these reasons, since the arrival in Shanghai could not be postponed, part of the trip was made with vehicles loaded on the support trucks.

The analysis of all tracks gives the total distance travelled in autonomous mode: in the whole trip 8244 km were covered in autonomous mode at an average speed of 38.4 km/h and a maximum speed of 70.9 km/h. A detail speed profile is reported in Fig. 34. The maximum distance covered in a day was 273 km and the maximum time in autonomous mode in a single day was 6 hours and 26 minutes.

A rough statistical pathplanner error using the route performed by the leader vehicle as ground-truth can also be provided. The histogram with the distribution of the differences between the paths made by the leader and the follower are shown in Fig. 35. As previously mentioned the difference between the two paths could also be given to the fact that the follower vehicle uses the leader trajectory as an indication only and merges it with the sensor data to generate optimal trajectories (e.g., to avoid an obstacle on the path).

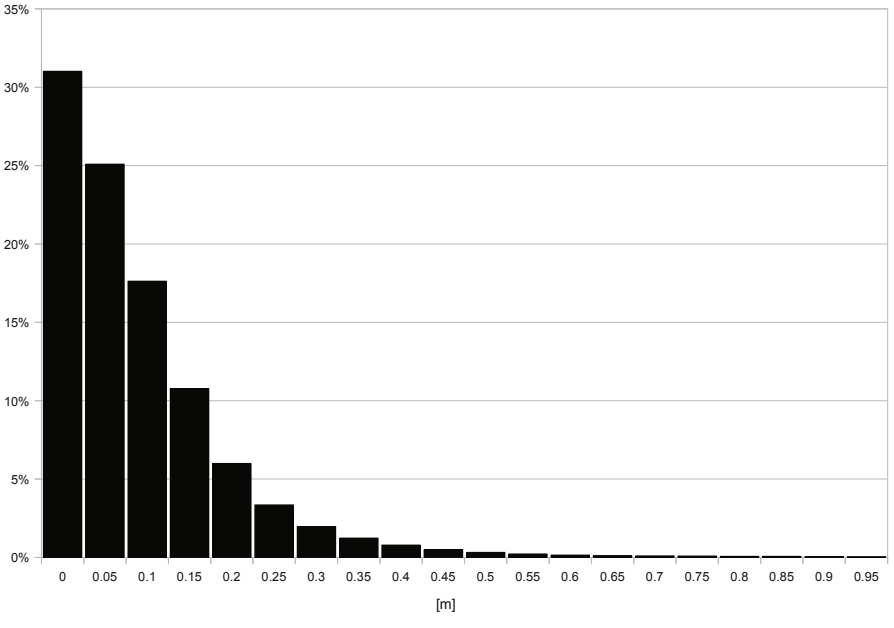


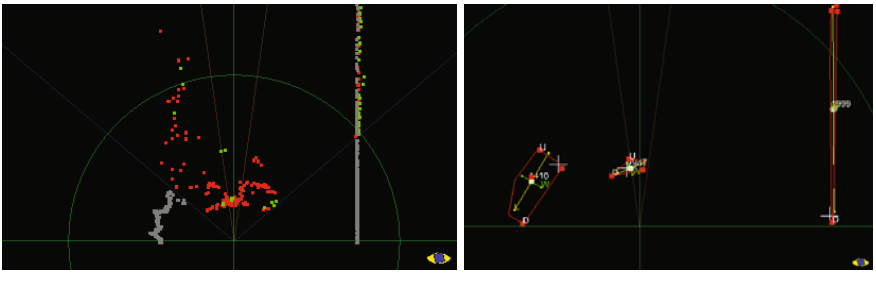
Fig. 35 Histograms of lateral crosstrack errors in autonomous Leader Vehicle Following mode generated by 8244 km of data analyzed.

5 Conclusions

The algorithms were tested during the whole VIAC expedition, under a wide variety of driving conditions and environments, such as chaotic traffic, heavy rain, fog, dust and unpaved roads. Some critical scenarios, detailed in the following, have emerged: the recorded data will be fundamental to further develop each algorithm and address the most complex situations.

5.1 Laser Obstacle Detection

Laserscanners are inherently sensitive to heavy rain and dust. Some devices, like the front LIDAR we installed on our vehicles, feature echos classification, with which it is possible to filter out rain and dust reflections. However, when the density of these elements become too high, laserscanners are no longer able to distinguish echos genereted by solid or by clouds of particles, consequently false detections might occur, as in Fig. 36.



(a) (b)

Fig. 36 Laser Obstacle Detection in case of dense dust: (a) echoes; (b) false detection

5.2 Lane Detection

Lane detection has produced robust results throughout the trip, even in heavy traffic conditions; nonetheless, some critical scenarios have been identified, such as a number of roads in asian countries where lane markings are characterized by distant and relatively short dashes, which are hard to correctly join together without producing false detections. Better integration with the IMU, allowing more predictable interest areas, is expected to improve the results.

5.3 Stereo Obstacle Detection

The obstacle detection algorithm has been designed making no assumptions on the scenarios to analyze, and this has proven a successful choice, since most situations are handled correctly. As explained in Sec 3.2.1, to reduce the number of missed detections to a minimum after the trip the stereo matching algorithm has been substituted with a SGM-based one, providing denser and more accurate maps.

5.4 Vehicle Detection

Vehicle Detection algorithm is sensitive to chaotic traffic, especially when other vehicles cut in between leader and follower vehicles. Even if the algorithm is able to

distinguish between a generic vehicle and the searched leader vehicle (analyzing colors, shape, position and size of rear light, etc., as explaied in Sec. 3.4.2) in these scenarios, typically, the leader is completely occluded and no detections are possible. Another critical situation is when a vehicle and another object, like a pedestrian or a bicycle, are fused together by the laser scanner: in this case the algorithm is no longer able to detect the vehicle shape, as explained in Sec. 3.4.1, and the leader detection may fail, like in Fig. 37.

(c)

Fig. 37 A vehicle and a pedestrian fused together by the Vehicle Detector

5.5 The Final Word

All in all the test was extremely successful since it provided the data that were needed. Although the test already ended and some conclusions can now be drawn on the systems' performance, the set of data acquired throughout the trip will be extremely valuable also in the coming future. It is estimated that they will provide an additional value to the improvement of the current algorithms and the design and prototyping of new ones for the next 2-3 years.

References

1. Bertozzi, M., Broggi, A., Fascioli, A.: Stereo Inverse Perspective Mapping: Theory and Applications. Image and Vision Computing Journal 8(16), 585–590 (1998)
2. Bombini, L., Cattani, S., Cerri, P., Fedriga, R.I., Felisa, M., Porta, P.P.: Test bed for Unified Perception & Decision Architecture. In: Procs. 13th Int. Forum on Advanced Microsystems for Automotive Applications, Berlin, Germany (May 2009)

3. Braid, D., Broggi, A., Schmiedel, G.: The TerraMax Autonomous Vehicle. Journal of Field Robotics 23(9), 693–708 (2006)
4. Braid, D., Broggi, A., Schmiedel, G.: The TerraMax Autonomous Vehicle concludes the 2005 DARPA Grand Challenge. In: Procs. IEEE Intelligent Vehicles Symposium 2006, Tokyo, Japan, pp. 534–539 (June 2006)
5. Broggi, A., Bertozzi, M., Fascioli, A., Conte, G.: Automatic Vehicle Guidance: the Experience of the ARGO Vehicle. World Scientific, Singapore (1999) ISBN 9810237200
6. Broggi, A., Cappalunga, A., Caraffi, C., Cattani, S., Ghidoni, S., Grisleri, P., Porta, P.P., Posterli, M., Zani, P.: TerraMax Vision at the Urban Challenge 2007. IEEE Trans. on Intelligent Transportation Systems 11(1), 194–205 (2010)
7. Broggi, A., Cerri, P., Ghidoni, S., Grisleri, P., Jung, H.G.: A New Approach to Urban Pedestrian Detection for Automatic Braking. IEEE Trans. on Intelligent Transportation Systems 10(4), 594–605 (2009) ISSN: 1524-9050
8. Chen, Y.-L., Sundareswaran, V., Anderson, C., Broggi, A., Grisleri, P., Porta, P.P., Zani, P., Beck, J.: TerraMax: Team Oshkosh Urban Robot. Journal of Field Robotics 25(10), 841–860 (2008)
9. Chen, Y.-L., Sundareswaran, V., Anderson, C., Broggi, A., Grisleri, P., Porta, P.P., Zani, P., Beck, J.: TerraMax: Team Oshkosh Urban Robot. In: Buehler, M., Iagnemma, K., Singh, S. (eds.) The DARPA Urban Challenge, Autonomous Vehicles in City Traffic. Springer Tracts in Advanced Robotics, pp. 595–622. Springer, Heidelberg (2009) ISBN: 978-3-642-03990-4
10. Felisa, M., Zani, P.: Incremental Disparity Space Image computation for automotive applications. In: Procs. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, St.Louis, Missouri, USA (October 2009)
11. Frchet, M.M.: Sur quelques points du calcul fonctionnel. Rendiconti del Circolo Matematico di Palermo (1884-1940) 22(1), 132–136 (1906)
12. Gehrig, S., Rabe, C.: Real-time semi-global matching on the cpu. In: ECVW 2010, pp. 85–92 (2010)
13. Hakimi, S.L., Schmeichel, E.F.: Fitting polygonal functions to a set of points in the plane. CVGIP: Graph. Models Image Process. 53(2), 132–136 (1991)
14. Hangouet, J.F.: Computation of the Hausdorff distance between plane vector polylines. In: Procs. Twelfth International Symposium on Computer-Assisted Cartography, Charlotte, Noth Carolina, USA, vol. 4, pp. 1–10 (1995)
15. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press (2004) ISBN: 0521540518
16. Hirschmüller, H.: Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. In: Intl. Conf. on Computer Vision and Pattern Recognition, San Diego, CA, USA, vol, June 2005, vol. 2, pp. 807–814. IEEE Computer Society Press, San Diego (2005)
17. Hirschmuller, H., Scharstein, D.: Evaluation of stereo matching costs on images with radiometric differences. PAMI 31(9), 1582–1599 (2009)
18. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence, vol. 2, pp. 674–679. Morgan Kaufmann Publishers Inc., San Francisco (1981)
19. Mallot, H.A., Bülthoff, H.H., Little, J.J., Bohrer, S.: Inverse perspective mapping simplifies optical flow computation and obstacle detection. Biological Cybernetics 64, 177–185 (1991)
20. Manduchi, R., Castano, A., Talukder, A., Matthies, L.: Obstacle detection and terrain classification for autonomous off-road navigation. Auton. Robots 18(1), 81–102 (2005)
21. McMaster, R.B.: A statistical analysis of mathematical measures for linear simplification. Cartography and Geographic Information Science 13(2), 103–116 (1986)

22. Nedevschi, S., Danescu, R., Schmidt, R., Graf, T.: High accuracy stereovision system for far distance obstacle detection. In: Procs. IEEE Intelligent Vehicles Symposium 2004, Parma, Italy (June 2004)
23. Nedevschi, S., Oniga, F., Danescu, R., Graf, T., Schmidt, R.: Increased Accuracy Stereo Approach for 3D Lane Detection. In: IEEE Intelligent Vehicles Symposium, Tokyo, Japan, pp. 42–49 (June 2006)
24. Peuquet, D.J.: An algorithm for calculating minimum euclidean distance between two geographic features. Computers & Geosciences 18(8), 989–1001 (1992)
25. Shi, J., Tomasi, C.: Good features to track. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 593–600 (January 1994)
26. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Comput. Surv. 38 (December 2006)