UNIVERSITY OF CALIFORNIA, SAN DIEGO

Learning from Local Image Regions

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Piotr Dollár

Committee in charge:

Professor Serge Belongie, Chair
Professor Gary Cottrell
Professor Sanjoy Dasgupta
Professor Truong Nguyen
Professor Zhuowen Tu
Professor Nuno Vasconcelos

2007

The dissertation of Piotr Dollár is approved, and it is acceptable in quality and form for publication on microfilm:

_____

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2007

To my parents.

TABLE OF CONTENTS

Signature Page . . . iii

Dedication . . . iv

Table of Contents . . . v

List of Figures . . . vii

Acknowledgements . . . x

Vita . . . xii

Abstract of the Dissertation . . . xiii

1 Introduction . . . 1

2 Supervised Learning of Edges and Object Boundaries . . . 4
- 2.1 Related work . . . 6
- 2.2 Problem formulation . . . 6
- 2.3 Learning edge probability . . . 8
  - 2.3.1 Features . . . 8
  - 2.3.2 Classification framework . . . 9
  - 2.3.3 Probabilistic Boosting Tree - training . . . 10
  - 2.3.4 Computing probability . . . 12
- 2.4 Experiments . . . 13
  - 2.4.1 Illustrations of Gestalt laws . . . 13
  - 2.4.2 Detecting object boundaries . . . 15
  - 2.4.3 Road detection . . . 18
  - 2.4.4 Edge detection in natural images . . . 18
- 2.5 Discussion . . . 23

3 Learning to Traverse Image Manifolds . . . 24
- 3.1 Related Work . . . 25
- 3.2 Algorithm . . . 26
- 3.3 Experiments on Point Sets . . . 30
- 3.4 Results on Images . . . 34
- 3.5 Discussion . . . 38
- 3.6 Mathematical Details . . . 39
  - 3.6.1 Equivalence of $\text{error}_1(\theta)$ and $\text{error}_2(\theta)$ . . . 40
  - 3.6.2 Minimization for General Manifolds . . . 41
  - 3.6.3 Minimization for Image Manifolds . . . 43
  - 3.6.4 LSML for General Manifolds . . . 45
  - 3.6.5 LSML for Image Manifolds . . . 46

4 Behavior Recognition via Sparse Spatio-Temporal Features 47
4.1 Related Work 49
4.2 Proposed Algorithm 52
4.2.1 Feature Detection 52
4.2.2 Cuboids 55
4.2.3 Cuboid Prototypes 58
4.2.4 Behavior Descriptor 59
4.3 Experiments 59
4.3.1 Datasets 59
4.3.2 Methodology 61
4.3.3 Algorithms for Comparison 63
4.3.4 Results 64
4.4 Discussion 66

Bibliography 69

LIST OF FIGURES

Figure 1.1: Local patch based representations have the advantage that they are robust to global transformations, occlusion, clutter, object and image variation, and so on, while retaining rich information about image content. Patches have found application in object detection ... 2

Figure 2.1: Manual segmentations of an example image and the corresponding edge maps. For illustration purposes, we show the negative of the probability so the darker the pixel, the higher the probability of an edge. 7
Figure 2.2: (A) Some positive (center pixel is an edge point) and (B) negative (center pixel is not an edge) image patches. 10
Figure 2.3: Learning the Gestalt laws of perceptual organization: parallelism. 14
Figure 2.4: Learning the Gestalt laws of perceptual organization: modal completion. 14
Figure 2.5: Learning the Gestalt laws of perceptual organization: alternate interpretation of the same data. 15
Figure 2.6: Illustration of object boundary detection. Two of fourteen training images are shown in the first row. The next rows show two of the seven testing images – `BEL` is able to generalize to the unseen images. Not only are boundaries of the mouse found 16
Figure 2.7: Zoomed in view of final testing image for mouse boundary detection and result of classifier. The classifier fails to detect edges near the moving head due to considerable motion blur, an artifact not observed in the training data. Two positive patches 17
Figure 2.8: Some results on road detection. Three satellite images were obtained from Google Maps, along with their corresponding road maps. The first two images were used for training, their corresponding road maps were converted to probability distributions of 19
Figure 2.9: Zoomed in view of a test image form the natural image dataset. Qualitative properties of our output can be seen: (1) `BEL` gives a true probability, not just a measure of edge strength. The strongest responses correlate well with regions where the largest number 20
Figure 2.10: Precision recall curves of `BEL` and `Pb` on gray (left) scale and color versions (right) of the Berkeley test set. The overall performance on gray scale images of `BEL` improves on the performance of `Pb`, whose performance is the highest reported in the literature... 21
Figure 2.11: The first row contains gray scale images from the Berkeley dataset (http://www.cs.berkeley.edu/projects/vision/grouping/segbench), and the second the overlayed manual segmentations. For each image we give our results and the result of the Berkeley 22

Figure 3.1: **Overview**. Twenty points (n=20) that lie on 1D curve (d=1) in a 2D space (D=2) are shown in (a). Black lines denote neighbors, in this case the neighborhood graph is not connected. We apply LSML to train $\mathcal{H}$ (with $f = 4$ RBFs). $\mathcal{H}$ maps points in $\mathbb{R}^2$ to 28
Figure 3.2: **Robustness**. LSML used to recover the embedding of the $\mathcal{S}$-curve under a number of sampling conditions. In each plot we show the original points along with the computed embedding (rotated to align vertically), correspondence is indicated by 31
Figure 3.3: **Reconstruction**. Reconstruction examples are used to demonstrate quality and generalization of $\mathcal{H}$. (a) Points sampled from the Swiss-roll manifold (middle), some recovered tangent vectors in a zoomed-in region (left) and embedding found by LSML (right). 33
Figure 3.4: **The translation manifold**. Here $F^i = X^i$; $s = 17$, $d = 2$ and 9 sets of 6 translated images each were used (not including the cameraman). (a) Zero padded, smoothed test image $\mathbf{x}$. (b) Visualization of learned Θ, see text for details. (c) $\mathcal{H}_\theta(\mathbf{x})$ computed 35
Figure 3.5: **Manifold generated by out-of-plane rotation of a teapot** (data from [77], sub-sampled and smoothed). Here, $d = 1$, $f = 400$ and roughly 3000 patches of width $s = 13$ were sampled from 30 frames. Bottom row shows the ground truth images; dashed box . . . 36
Figure 3.6: **Traversing the eye manifold**. LSML trained on one eye moving along five different lines (3 vertical and 2 horizontal). Here $d = 2$, $f = 600$, $s = 19$ and around 5000 patches were sampled; 2 frames were considered neighbors if they were adjacent in time 37

Figure 4.1: Visualization of cuboid based behavior recognition. A spatio-temporal volume of mouse footage shown at top. We apply a spatio-temporal interest point detector to find local regions of interest in space and time (cuboids) which serve as the substrate for behavior. . . 48
Figure 4.2: Example of six cuboids extracted from two different sequences of grooming. A single frame is shown from each original sequence. Below, each cuboid is shown over time. Note that although the posture of the mouse is quite different in the two cases. 50
Figure 4.3: Filters used to detect interest points, tuned to fire maximally when intensity in a local spatial region oscillates temporally. Dark lobes correspond to negative areas. Surfaces shown are drawn at 10% of the peak filter response. The 1D profile of the temporal filters . . . 55
Figure 4.4: Shown is the intra and inter class performance of our recognition method on the face dataset using different cuboid descriptors. The full algorithm, dataset and methodology are discussed later, the sole purpose of this figure is to give a sense of the relative. 57
Figure 4.5: Representative frames from clips in each domain: (a) facial expressions, (b) mouse behavior, and (c) human activity. 60

Figure 4.6: We tested how sensitive the performance of our method is to various parameter settings on the face dataset. In each of the above curves we plot classification error for 10 different settings of a given parameter with all other parameters kept constant at default, ... 62
Figure 4.7: FACE DATASET *Top row:* We investigated how identity and lighting affect each algorithm's performance. In all cases CUBOIDS gave the best results. EFROS and CUBOIDS+HARRIS had approximately equal error rates, except that EFROS tended to perform 65
Figure 4.8: MOUSE DATASET *Left:* Confusion matrix generated by CUBOIDS on the full mouse dataset. As mentioned, this dataset presents a challenging recognition problem. Except for a few difficult categories, recognition rates using our method were fairly high. *Right:* 66
Figure 4.9: HUMAN ACTIVITY DATASET Shown are confusion matrices generated by CUBOIDS. Two classifiers were used: 1-nearest neighbor and Support Vector Machines with radial basis functions [32]. Using SVMs resulted in a slight reduction of the error. Note that most . . . 67

ACKNOWLEDGEMENTS

When entering graduate school, I only had a vague idea of what I would accomplish. Serge Belongie gradually but effectively steered me in the right direction and helped me formulate my vision and goals as a scientist. His help both on specific research topics and on coping with the daily challenges facing a scientist has been invaluable. Serge was also always very supportive; I could not ask more from an advisor.

Zhuowen Tu has greatly influenced how I approach real world problems and how to see a project through from beginning to end. He has shared practical knowledge as well as his wisdom and philosophical insight into computer vision. Throughout our work together Zhuowen has kept my passion for research alive, especially during the tough times that we must all inevitably face. In short, I was lucky enough not to have just one great advisor but two.

I would also like to thank others who helped guide my research at UCSD. Sanjoy Dasgupta is one of the best teachers I've met. Also, somewhat surprisingly, he never seemed to get tired of my pestering him with random problems and wild theories. Charles Elkan has been supportive and insightful, and helped teach me how to present myself and my work. I would also like to thank Gary Cottrell for his support and guidance in my research, especially in the early phases of my studies, and Nuno Vasconcelos for insightful conversations.

Graduate school would have been much lonelier without the other students who shared the same plight as I. Vincent Rabaud has played a very significant role in more than half of the research I have done. He has also kept me happy and was a friend – the number of sleepless nights we spent together coding or hacking away at some particularly hairy math without killing each other is surprising. Kristin Branson has been very supportive and helped put many of my earlier ideas on more solid ground. Sameer Agarwal and Eric Wiewiora were excellent resources who know far too much for their own good. Working and sharing my excitement of computer vision with Boris Babenko has been great. Many others have made these years enjoyable – including Lawrence Cayton, Ben Ochoa, Craig Donner, Andrew Rabinovich, Doug Turnbull, Luke Barrington, Matt Tong, Lingyun Zhang, Carolina

Galleguillos, Stephan Steinbach, Josh Wills, Will Chang and Tim Marks among others.

One person that deserves my special thanks is Anna Shemorry. Her friendship and her love make everything I do a little more worthwhile. Her patience is almost uncanny – not only did she accept my sleeping in lab a week at a time before a deadline, she would also bring me food and occasionally a change of clothes. I hope I can repay my debt to some small extent as Anna pursues her own doctorate.

Finally, my parents' attention and guidance have made me the person I am today, from engrossing me with math puzzles for hours at a time to pushing me to always challenge myself. Against my best efforts, some of their knowledge has rubbed off on me. I can only hope to make them proud.

Portions of this dissertation are based on papers that I have co-authored with others. Listed below are my contributions to each of these papers.

1. Chapter 2 is in part based on the paper "Supervised Learning of Edges and Object Boundaries" by P. Dollár, Z. Tu and S. Belongie [17]. The dissertation author was responsible for implementing the algorithm, performing the experiments and writing the paper.

2. Chapter 3 is in part based on the paper "Learning to Traverse Image Manifolds" by P. Dollár, V. Rabaud and S. Belongie [15]. The dissertation author proposed the initial idea, developed much of the mathematics and code, and wrote the bulk of the paper.

3. Chapter 4 is in part based on the paper "Behavior Recognition via Sparse Spatio-Temporal Features" by P. Dollár, V. Rabaud, G. Cottrell and S. Belongie, [16]. The dissertation author was responsible for the development of the algorithm, experimental design and literature survey, and also wrote most of the paper.

VITA

1980	Born, Krakow, Poland.
2002	A. B., Harvard University.
2002	S. M., Harvard University.
2007	Ph. D., University of California, San Diego.

PUBLICATIONS

B. Babenko, P. Dollár and S. Belongie, "Task Specific Local Region Matching," *IEEE International Conference on Computer Vision* (**ICCV**), 2007.

P. Dollár, V. Rabaud and S. Belongie, "Non-Isometric Manifold Learning: Analysis and an Algorithm," *International Conference on Machine Learning* (**ICML**), 2007.

P. Dollár, Z. Tu, H. Tao and S. Belongie, "Feature Mining for Image Classification," *IEEE Conference on Computer Vision and Pattern Recognition* (**CVPR**), 2007.

P. Dollár, V. Rabaud and S. Belongie, "Learning to Traverse Image Manifolds," *Neural Information Processing Systems* (**NIPS**), 19, 2006.

P. Dollár, Z. Tu and S. Belongie, "Supervised Learning of Edges and Object Boundaries," *IEEE Conference on Computer Vision and Pattern Recognition* (**CVPR**), 2006.

P. Dollár, V. Rabaud, G. Cottrell and S. Belongie, "Behavior Recognition via Sparse Spatio-Temporal Features," *IEEE International Conference on Computer Vision - Visual Surveillance and Performance Evaluation of Tracking and Surveillance* (**ICCV VS-PETS**), 2005.

S. Belongie, K. Branson, P. Dollár, and V. Rabaud, "Monitoring Animal Behavior in the Smart Vivarium," *Measuring Behavior*, 2005.

P. Dollár, P. Laskowski and M. Van Alstyne, "Simulating the Growth and Diffusion of Knowledge in Agent Societies," *Conference on Computational Analysis of Social and Organization Systems* (**CASOS**), 2002.

O. Chmaissem, J.D. Jorgensen, H. Shaked, P. Dollár, J.L. Tallon, "Crystal and magnetic structure of ferromagnetic superconducting $RuSr_2GdCu_2O_8$," *Physics Review B*, 61, 2000.

ABSTRACT OF THE DISSERTATION

Learning from Local Image Regions

by

Piotr Dollár
Doctor of Philosophy in Computer Science
University of California San Diego, 2007
Professor Serge Belongie, Chair

A trend in computer vision over the last decade or so has been to describe the statistics and content of images in terms of local image regions, *i.e.*, image patches. Applications have included object detection, scene recognition, texture classification and image categorization. Local patch based representations have the advantage that they are robust to global transformations, occlusion, clutter, object and image variation, and so on, while retaining rich information about image content. This is the case even when global information relating the relative position of patches is not used, as in so called "bags of words" approaches. Furthermore, in the supervised learning framework where labeled images are a source of data, characterizing images using patches means a single image can provide a large number of patches for training. These properties suggest local patch based representations should continue to find expanded use in computer vision.

In this dissertation we show the application of patch based methods to three domains for which traditionally more global approaches have been used. First we show how the classic problem of edge detection can be posed as a series of patch by patch decisions that can be solved in a supervised learning framework. We show the application of this approach to a number of specific domains such as mouse boundary detection and road detection. Second, we show how modeling object warps and highly non-linear image transformations can again be done locally, thus avoiding computational challenges and the scarcity of data typically associated with

these problems. For example, our approach is able to learn eye motion and out-of-plane rotation of a teacup from sparse data. Third, we extend the notion of local regions from 2D to 3D, i.e. from patches to cuboids, in order to model the content of video. We show applications to behavior recognition in a number of domains including human activity and mouse behavior.

The methods we introduce here advance the state of the art and have the potential to be useful in a broad range of applications in computer vision. Our approach to edge detection currently outperforms all competing approaches for gray scale edge detection and comes in close second for color edge detection on the well established Berkeley Segmentation Dataset [48]. We hope it will play a similar role as Canny edge detection but for highly textured, real world images. Our approach to modeling object warps locally showed dramatic improvements over previous such methods [4], and helped solidify the theoretical foundation of nonlinear manifold learning. Finally, our cuboids formalism is simple yet powerful, and has already been utilized in two vision systems [54, 79]. It has the potential to serve as the basis for a broad range of methods for describing the contents of video. Overall, our contribution has been to help establish the importance of patch based approaches and to expand our understanding of a fundamental aspect of computer vision.

1

Introduction

A trend in computer vision over the last decade or so has been to describe the statistics and content of images in terms of local image regions, *i.e.*, image patches. Applications have included object detection, scene recognition, texture classification and image categorization. Local patch based representations (see Figure 1.1) have the advantage that they are robust to global transformations, occlusion, clutter, object and image variation, and so on, while retaining rich information about image content. This is the case even when global information relating the relative position of patches is not used, as in so called "bags of words" approaches. Furthermore, in the supervised learning framework where labeled images are a source of data, characterizing images using patches means a single image can provide a large number of patches for training. These properties suggest local patch based representations should continue to find expanded use in computer vision.

In this dissertation we show the application of patch based methods to three domains for which traditionally more global approaches have been used, including edge detection, manifold learning and behavior recognition.

First we show how the classic problem of edge detection can be posed as a series of patch by patch decisions that can be solved in a supervised learning framework. Edge detection is one of the most studied problems in computer vision, yet it remains a very challenging task. It is difficult since often the decision for an edge cannot be made purely based on low level cues such as the gradient, instead we need to engage all levels of information, low, middle, and high, in order to decide

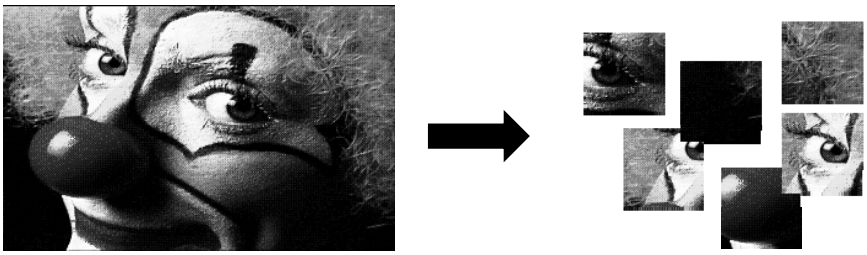


Figure 1.1: Local patch based representations have the advantage that they are robust to global transformations, occlusion, clutter, object and image variation, and so on, while retaining rich information about image content. Patches have found application in object detection, scene recognition, texture classification and image categorization. In this work we show how patch based approaches can also be used for edge detection, manifold learning and behavior recognition.

where to put edges. In Chapter 2 we propose a novel supervised learning algorithm for edge and object boundary detection which we refer to as Boosted Edge Learning or `BEL` for short. A decision of an edge point is made independently at each location in the image; a very large aperture is used providing significant context for each decision. In the learning stage, the algorithm selects and combines a large number of features across different scales in order to learn a discriminative model using an extended version of the Probabilistic Boosting Tree classification algorithm. The learning based framework is highly adaptive and there are no parameters to tune. We show applications for edge detection in a number of specific image domains, such as mouse boundary detection and road detection, as well as on natural images. We test on various datasets including the Berkeley dataset and the results obtained are competitive with he best results in the literature.

In Chapter 3 we present a new algorithm, Locally Smooth Manifold Learning (LSML), that learns a warping function from a point on a manifold to its neighbors. We show how modeling object warps and highly non-linear image transformations can again be done locally, thus avoiding computational challenges and the scarcity of data typically associated with these problems. The key insight to working with images is that although images can live in very high dimensional spaces, we do not have to learn transformations with that many parameters. Essentially, we

assume that each pixel in an image transformation can be computed only using the information in a patch centered on the corresponding pixel. The resulting technique scales independently of the number of pixels in an image, furthermore different sized images can be used. The per patch assumption is not always suitable, most notably for transformations that are based only on image coordinates and are independent of appearance; however in many cases our technique results in a very effective representation. Among other applications, we show our approach is able to learn eye motion and out-of-plane rotation of a teacup from sparse data.

A common trend in object recognition is to detect and leverage the use of sparse, informative feature points. The use of such features makes the problem more manageable while providing increased robustness to noise and pose variation. In Chapter 4, we extend the notion of local regions from 2D to 3D, *i.e.*, to the spatio-temporal case. For this purpose, we show that the direct 3D counterparts to commonly used 2D interest point detectors are inadequate, and we propose an alternative. Anchoring off of these interest points, we devise a recognition algorithm based on spatio-temporally windowed data. Recognition results on a variety of datasets including both human and rodent behavior are presented.

2

Supervised Learning of Edges and Object Boundaries

Edge detection is one of the most studied problems in computer vision. It finds application in tasks such as object detection/recognition, structure from motion, segmentation and tracking, e.g. [68, 70, 9, 75]. Edges reduce the dimensionality of the original data while retaining rich information about the contents of the image. They can also serve as a basis for other forms of image representation, such as the primal sketch [47, 30]. Nevertheless, high quality general edge detection remains elusive. Methods that rely on local features, such as the Canny [12] detector, do not take into account the context (e.g. if the surrounding area is textured), mid-level information (e.g. the Gestalt laws [41]), or high level information (e.g. object knowledge [74]). Canny also cannot take into account local information at multiple scales. Such information is important: sometimes we hallucinate a boundary where there is weak or even no local evidence (e.g. certain parts of an object may have the same intensity pattern as the background), other times we do not see a boundary even if there are strong local cues that would imply its existence (e.g. in the presence of shadows). Complex generative models, such as presented in [74, 58], have the potential to integrate both low-level and high-level information but present significant computational challenges.

Even as intensive research into general edge detection continues, there is general consensus in the community that edge detection is somewhat ill defined in

that it is not quite clear what defines a correct output [48]. From an application driven point of view, a general edge detection algorithm is possibly inappropriate since relevant boundaries in a scene depend on the components of interest, which in turn depend on the task being performed. For example, if the goal is to detect object boundaries for object detection then all other detected edges are noise, but if the task changes the object boundaries may no longer be relevant. One of the motivations for this chapter is that while designing individual edge detectors for particular tasks in particular domains is not feasible, often times it is simple to obtain training images with labeled boundaries, and depending on the task there need not be any ambiguity as to what constitutes a correct output.

In this chapter we propose a novel supervised learning algorithm for edge and object boundary detection which we refer to as Boosted Edge Learning or `BEL` for short. A decision of an edge point is made independently at each location in the image; a very large aperture is used providing significant context for each decision. In the learning stage, the algorithm selects and combines a set of features out of a pool with tens of thousands of generic, efficient Haar wavelets in order to learn a discriminative model. The scope of the machine learning problem is formidable: we have tens of millions of training points with tens of thousands of features each. We present an extension of the probabilistic boosting tree algorithm that copes with this data much better than did either boosting or cascade approaches [76]. Our method outputs true probabilities, whereas other edge detection methods either output a binary value or a soft value based on edge strength (which is not a true probability). We show how the method implicitly combines low-level, mid-level, and context information across different scales when making a decision. The learning based framework is highly adaptive and there are no parameters to tune. We show applications for edge detection in a number of specific image domains as well as on natural images. We test on various datasets including the Berkeley dataset and the results obtained are very good.

2.1 Related work

There have been many methods proposed for edge detection, we have already mentioned a few. A representative set is given by [12, 55, 74, 58], we review each in some detail below. The Canny edge detector [12] is perhaps the most widely used edge detector; it is based only on local gradient and has a scale parameter to tune. Ramesh [55] systematically analyzed the performance of a number of edge detectors w.r.t. their parameter setting. The method in [74] uses edges obtained from bottom-up (discriminative) processes as proposals to guide the top-down (generative) search. Recently, Ren et al. [58] build graphs to reinforce some mid-level cues to give more complete edges. In general, however, it is difficult to encode all the rules needed for edge detection, for example how to exploit all sorts of mid-level Gestalt laws such as junction, parallelism, symmetry, and closure, how to deal with texture and color, how to resolve disagreements between cues that give conflicting local information, and so on, even if mid-level and high-level information can be modeled (for example in a generative framework), a search for optimal solutions can be daunting.

Two other relevant algorithms are `Pb`, proposed by [48], and the method presented in [42]. Both have data driven and learning components, although aside from this obvious similarity they bear little resemblance to our approach. `Pb` uses learning to perform cue combination on 9 carefully designed local features (texture gradient, brightness gradient and color gradient at 3 scales each), learning improves performance over setting the weights by hand. The learning component in both [48] and [42] improves the overall results of the algorithms on general edge detection; our method, however, relies entirely on learning. This makes our method very versatile, and as we show in the experiments section we were able to apply it to a very broad range of domains.

2.2 Problem formulation

A number of tasks in computer vision can be formulated as finding a likely interpretation W for an observed image $\mathbf{I}$, where W includes information about the spatial location and extent of objects, regions, object boundaries, curves and so

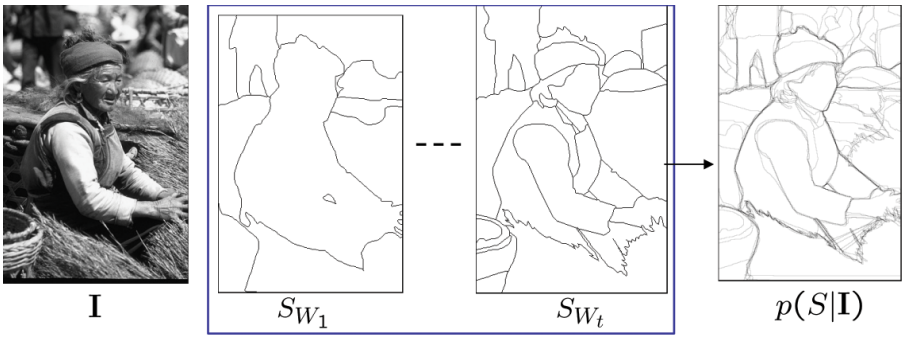


Figure 2.1: Manual segmentations of an example image and the corresponding edge maps. For illustration purposes, we show the negative of the probability so the darker the pixel, the higher the probability of an edge.

on. Let S_W be a function associated with a scene interpretation W that encodes the spatial location and extent of a component of interest, where $S_W(i,j)$ is 1 for each image location (i,j) that belongs to the component and 0 elsewhere. Given an image, obtaining an optimal or even likely scene interpretation W, or associated S_W, can be difficult. Instead, we can ask what is the probability a given location in a given image belongs to the component of interest:

$$p(S(i,j)|\mathbf{I}) = \sum_{W_t} S_{W_t}(i,j)p(W_t|\mathbf{I}), \tag{2.1}$$

where t indexes each individual's interpretation. See Figure (2.1). Calculating $p(S(i,j)|\mathbf{I})$ using the above is difficult. Instead we seek to learn this distribution directly from image data. To further reduce the complexity, we seek to learn a discriminative model $p(S(i,j)|\mathbf{I}_{N(i,j)})$ where $\mathbf{I}_{N(i,j)}$ is an image patch centered at (i,j). If we throw away the absolute coordinates, then the major focus of this chapter is to learn:

$$p(S(c)|\mathbf{I}_{N(c)}), \tag{2.2}$$

where c is the center of an image patch. So the decision for a single point is made based on an image patch centered at it. A large enough patch contains low-level features and also some mid-level and context information. We want to piece this information together and learn a discriminative model.

Edge detection is easily placed in the above framework if we let the scene interpretation W of an image be its segmentation, and define $S_W(i,j) = 1$ if any region boundary in W passes through location (i,j), and otherwise $S_W(i,j) = 0$. Then $p(S(i,j)|\mathbf{I})$ gives the edge probability at each location in the image. To obtain ground truth for our discriminative model, we use the manually labeled segmentations from [48]. Given an image and a number of different segmentations W_t, we can obtain an approximation, $\hat{p}(S|\mathbf{I})$, by considering each of the human segmentations to have an equal probability $p(W_t|\mathbf{I})$. We can then sample positive and negative example patches from $\mathbf{I}$ according to $\hat{p}(S|\mathbf{I})$. Figure (2.1) shows an example where human subjects drew different segmentations. The edge probability map summing up all manual segmentations is shown on the right. Figure (2.2) shows some sample patches.

We describe other applications of this framework, as well as other methods for generating the ground truth, in the experiments Section 2.4. However, for much of this chapter we refer to $p(S(c)|\mathbf{I}_{N(c)})$ as the edge probability at c.

2.3 Learning edge probability

Our goal is to train a discriminative model $p(S(c)|\mathbf{I}_{N(c)})$ that predicts the probability of a location being an edge point based on an image patch centered at it.

2.3.1 Features

Informative features greatly facilitate the training stage of a classification algorithm. Their design should be a compromise among generality, speed and effectiveness. As mentioned, `Pb` performs cue combination on 9 features. These features are a culmination of years of effort, and they are in fact very effective for edge detection in natural images. Instead, our approach is to use tens of thousands of very simple features, calculated over a much larger image region. The primary advantage of such an approach is that the human effort is minimized, instead the work is shifted to the classification algorithm. We can measure the time it took to design and implement our features in terms of days. Also, such features tend to be much

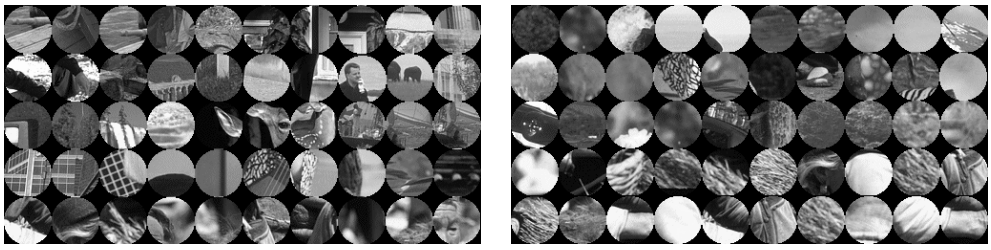
more general so applying the algorithm to a different domain is straightforward. Including features far from the center of the patch implicitly provides mid-level and context information to the discriminative algorithm. The primary disadvantage is that the classification stage is far more challenging and the choice of a discriminative algorithm more sensitive.

We used a large number of generic features at multiple locations, orientations, scales, aspect ratios and so on, calculated over a large image patch (e.g. of size 50×50). Features included gradients at multiple scales and locations, differences between histograms computed over filter responses (difference of Gaussian (DoG) and difference of offset Gaussian (DooG)) again at multiple scales and locations, and also Haar wavelets[76]. We experimented with using the output of the Canny edge detector at various scales as input to our method, although these Canny features were not very informative and for speed reasons were not included in the final version of our classifier. The classifier has to handle edges at different scales implicitly, and also different types of edges, so it is important to have a large pool of informative features. Integral images, filter responses, and so on were computed once for each input image, not once per patch, increasing efficiency when adjacent patches must be evaluated. For color images we additionally calculated the above features (gradients, histograms of filter responses, Haar wavelets) over each color channel.

We use approximately 50000 features. The same features were used in all applications reported. We emphasize that little effort went into optimizing our feature set.

2.3.2 Classification framework

Given a training image, along with an estimate of the probability that each location is an edge point, $\hat{p}(S|\mathbf{I})$, we can sample positive and negative example patches. The number of samples from a single image is equal to the number of locations in the image (although typically the majority of locations contain negative samples), and even more if we consider the image at multiple orientations, scales and so on. Adjacent samples are highly similar, nevertheless, we often have very large training sets ($\mathcal{O}(10^8)$ samples). Learning an accurate decision boundary for this data is difficult, and we would like to use as much data as possible.



(A) Positives (B) Negatives

Figure 2.2: (A) Some positive (center pixel is an edge point) and (B) negative (center pixel is not an edge) image patches.

Advances in machine learning have allowed us to take advantage of the size of this high dimensional dataset. Boosting [23, 24] deals well with high dimensional data. Cascades of boosted classifiers [76] allow for efficient evaluation, and combined with bootstrapping allow for training on very large datasets. In an early attempt we trained a cascade of AdaBoost classifier, unfortunately the cascade did not give us sufficiently good results. Even with a large number bootstrapping stages the error remained fairly high (a comparison is given in the experiments section, specifically see Figure 2.10).

Instead we chose to use an extension of the probabilistic boosting tree (`PBT`) proposed in [73]. `PBT` can be seen as a combination of a decision tree with boosting (a cascade is then just a special case of a tree). In this chapter, we give an extended `PBT` that combines the bootstrapping procedure directly into the tree formation while properly maintaining priors. We give a description of the algorithm below, our presentation of the extended `PBT` algorithm uses a different notation from [73].

2.3.3 Probabilistic Boosting Tree - training

Training `PBT` is similar to training a decision tree, except at each node a boosted classifier is used to split the data. The tree is trained recursively: at each node the empirical distribution $\hat{q}(y)$ of the data is calculated, and if the node is not pure ($0 < \hat{q}(y) < 1$), a strong classifier is trained on the data at the node. Each sample is then passed to the left and right subtrees, weighted by $q(-1|x_i)$ and

$q(+1|x_i)$ respectively, where $q(+1|x_i)$ is the probability that x_i is a positive sample according to the strong classifier. Thus, the strong classifier at each node is used not to return the class of the sample but rather to assign the sample to the left or right subtree. Training proceeds recursively. Details for the extended version of the algorithm are given below, see [73] for information about the original algorithm. To train:

1. Given a set of images with edges annotated, retrieve a training set

$$S = \{(x_1, y_1, w_1), ..., (x_m, y_m, w_m)\ x_i \in \chi,\ y_i \in \{-1, +1\},$$

   where $\sum_i w_i = 1$.

2. If the number (or weight) of either positive or negative samples in S is too small, perform bootstrapping to augment S (see below).

3. Compute the empirical distribution of S, $\hat{q}(y) = \sum_i w_i \delta(y_i = y)$. Continue if the depth of the node does not exceed some maximum value and $\theta \leq \hat{q}(+1) \leq (1 - \theta)$, e.g. $\theta = 0.99$, else stop.

4. On training set S, train a strong boosted classifier (with a limited number of weak learners). The number of weak classifiers is set by hand, alternatively it could be set by using cross validation.

5. Split the data into two sets S_L and S_R using the decision boundary of the learned classifier and a tolerance ϵ. For each sample (x_i, y_i, w_i) compute $q(+1|x_i)$ and $q(-1|x_i)$, then:
   $(x_i, y_i, w_i * q(+1|x_i)) \rightarrow S_R$
   $(x_i, y_i, w_i * q(-1|x_i)) \rightarrow S_L$.
   Finally normalize all the weights in S_L and also S_R.

6. Train the left and right children recursively using S_L and S_R respectively (go to step 2).

The bootstrapping step (2) for a given node is similar to bootstrapping when training a cascade, except that both positive and negative examples are bootstrapped. Let $l_1, ...l_k$ denote the path to the current node at depth $k + 1$. The

weight of a sample (x, y, w) when it reaches the node is given by $w' = w \prod q(x|l_i)$. By resampling the original data after reweighing according to the above (and renormalizing), one can augment the data S at the given node. This bootstrapping procedure allows us to deal with very large data sets while properly maintaining priors.

Finally, to make training more efficient, step (5) can be altered so a given sample is passed to both S_L and S_R only if it is near the decision boundary (the bootstrapping procedure must be altered accordingly):

If $q(+1|x_i) - \frac{1}{2} > \varepsilon$:
  $(x_i, y_i, w_i) \rightarrow S_R$
else if $q(-1|x_i) - \frac{1}{2} > \varepsilon$:
  $(x_i, y_i, w_i) \rightarrow S_L$
else:
  $(x_i, y_i, w_i * q(+1|x_i)) \rightarrow S_R$
  $(x_i, y_i, w_i * q(-1|x_i)) \rightarrow S_L$.

2.3.4 Computing probability

Given a trained tree, the posterior $\tilde{p}(y|x)$ is computed recursively. If the tree has no children, the posterior is simply the learned empirical distribution at the node $\tilde{p}(y|x) = \hat{q}(y)$. Otherwise the posterior is defined recursively:

$$\tilde{p}(y|x) = q(+1|x)\tilde{p}_R(y|x) + q(-1|x)\tilde{p}_L(y|x)$$

Here $q(y|x)$ is the posterior of the classifier, and $\tilde{p}_L(y|x)$ and $\tilde{p}_R(y|x)$ are the posteriors of the left and right trees.

In other words, to compute the posterior at a non-terminal node of the tree, the posterior of the left and right subtrees are calculated and the resulting posterior is a weighted combination according to the output of the strong classifier associated with the given node. Just as in training, we avoid traversing the entire tree by recursing to both subtrees only if the sample is near the decision boundary:

If $q(+1|x) - \frac{1}{2} > \varepsilon$:
  $\tilde{p}(y|x) = q(+1|x)\tilde{p}_R(y|x) + q(-1|x)\hat{q}_L(y)$
else if $q(-1|x) - \frac{1}{2} > \varepsilon$:
  $\tilde{p}(y|x) = q(+1|x)\hat{q}_R(y) + q(-1|x)\tilde{p}_L(y|x)$

else:
$\tilde{p}(y|x) = q(+1|x)\tilde{p}_R(y|x) + q(-1|x)\tilde{p}_L(y|x)$

Typically, using this approximation, only a few paths in the tree are traversed; thus the amount of computation to calculate $\tilde{p}(y|x)$ is roughly linear in the depth of the tree.

2.4 Experiments

We show the application of our framework to 4 different domains: (1) illustrations of the Gestalt laws, (2) detection of object boundaries, (3) road detection and (4) edge detection in natural images. We use nearly identical parameters in all domains, except we change the depth of the tree depending on the amount of data available to avoid overfitting.

2.4.1 Illustrations of Gestalt laws

The Gestalt laws of perceptual organization, including symmetry, closure, parallelism and so on, are rules of how component parts are organized into overall patterns [41]. The Gestalt laws play an important role in determining object grouping and region boundaries. Explicit studies of mid-level structures include their representations, spatial relationships, and explicit probability distributions. Though there has been substantial work done in this vein [18, 27, 30, 58, 78], individual studies tend to focus on particular Gestalt laws, and it is not clear how to combine them into a unified framework.

Typically, applying the Gestalt laws is seen as a separate 'mid-level' processing stage that comes into play after low-level features have been calculated. Instead, we show how in our framework the Gestalt laws can be exploited implicitly in a discriminative model that uses very simple image features. The advantages to this type of approach are as follows. First, providing training data is simple – we just need to annotate the edges where the Gestalt laws apply. The same training process can then be used. We do not have to define the individual laws, or how they interact, instead the learning algorithm combines a set of features/weak classifiers to generalize based on the training samples.

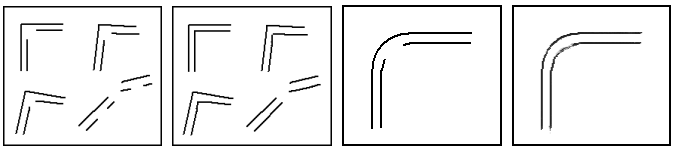


Figure 2.3: Learning the Gestalt laws of perceptual organization: parallelism.

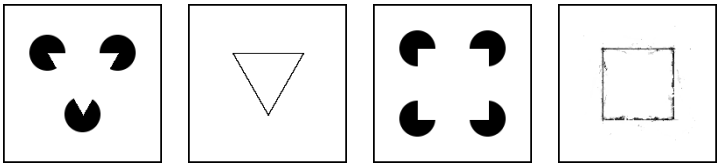
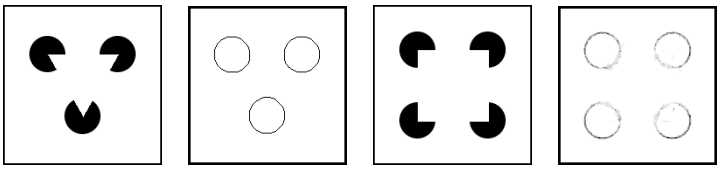


Figure 2.4: Learning the Gestalt laws of perceptual organization: modal completion.

In the remainder of this section we provide a few of examples in the form of analogies: we provide a training image and the annotated ground truth, train and apply the discriminative model to a novel test image. That is, we ask "A is to B as C is to ?", akin to the work of [34]. The input images in these examples are binary. The results for these examples were easy to obtain – they required no special parameter settings or tweaks to the algorithm. These are meant to be illustrative, we argue that on real data similar principles are implicitly exploited to achieve good results.

Parallelism, see Figure 2.3. Training is done on the image in (A) with the ground truth given in (B), and the result when the classifier is applied to (C) is given in (D). To correctly classify points as edge points the local gradient is insufficient. Instead, some edges must be filled in using 'parallelism'. Note that on the test image the learned classifier outputs a high edge probability not only in locations where local gradient was present but also in the gap. The learned classifier was able to generalize to two parallel curves with a smooth turn from training data that contained only straight parallel lines.

Modal completion, see Figure 2.4. Modal completion occurs when portions



(A) (B) (C) (D)

Figure 2.5: Learning the Gestalt laws of perceptual organization: alternate interpretation of the same data.

of an object are occluded by another object that happens to have the same color as nearby regions [66]. Shown the so called Kanizsa triangle in figure (A), observers typically report a white triangle occluding three black disks. In cases such as this, there is a perception of a contrast border even though there is no local contrast. Our algorithm is easily able to generalize to the example of the square. This is significant because it shows that the method can hallucinate edges by fusing features on a relatively large image neighborhood. Note that edge hallucination is not possible with the algorithm of Martin et al. [48], where only local gradients are available.

The same data can have an alternate interpretation, see Figure 2.5. We are agnostic with respect to preconceived notions about what defines an edge. The algorithm learns based on the training data it is given.

2.4.2 Detecting object boundaries

We can train the algorithm to detect edges specifically between an object and the rest of the image. By training on object boundaries the discriminative model learns to suppress other edges in the image, regardless of local gradients.

If we let the scene interpretation W of an image include the location and extent of an object of interest, and define $S_W(i,j) = 1$ if the object boundary passes through location (i,j), then $p(S(i,j)|\mathbf{I})$, defined as before, gives the probability that each location in the image is on the boundary of the object of interest. To obtain ground truth we roughly outlined the object in each image using a paint program. Since the labeling was error prone we smoothed the binary edge map using a Gaussian kernel, which assumes a Gaussian model of the localization error. The result was an approximation of the probability that each location contains an

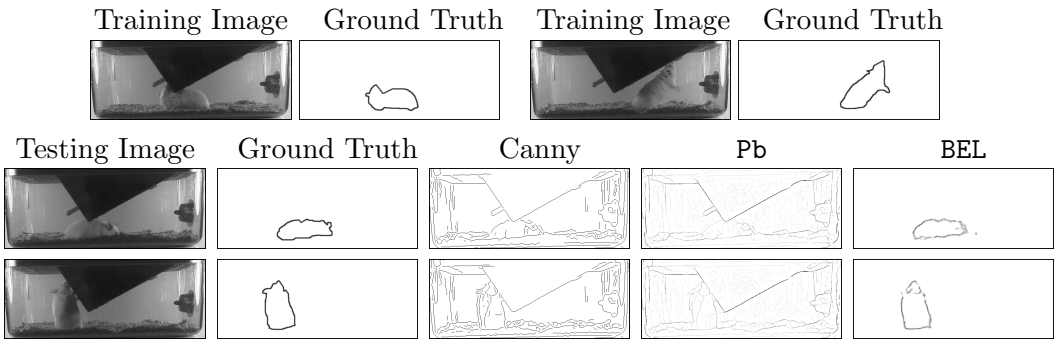


Figure 2.6: Illustration of object boundary detection. Two of fourteen training images are shown in the first row. The next rows show two of the seven testing images – BEL is able to generalize to the unseen images. Not only are boundaries of the mouse found, but also other edges are suppressed – something that traditional edge detection algorithms cannot do. Both Canny and Pb perform poorly, the 'F' score for BEL was .79 while for Canny it is .10 and for Pb it is .13 (for more on evaluation methodology see Section 2.4.4). Not only do Canny and Pb detect non-object edges but detection of the mouse edges is poor.

edge, $\hat{p}(S(i,j)|\mathbf{I})$, as before.

Results are shown in Figure (2.6). Results on the testing data match the human labeled images well, although in some parts the mouse edges were not detected (especially around the ears and tail). Very few non-object edges were detected. The results are far superior to both Canny and Pb [48] which are not tuned for this specific domain. Although Pb has a data driven component, code for training is not available online, furthermore even if retrained we would not expect Pb to suppress non-object edges since only local gradients are available as features.

In Figure (2.7) we show a closeup of the final testing image, and two cropped patches, that demonstrate the importance of using significant context information.

Successful detection of object boundaries can facilitate object detection or tracking. Ten images containing a given object provide only ten instances of the object, yet tens of thousands of points on the boundaries of the object (although these points may be highly correlated). Detecting the general location of the mouse in the images in Figure (2.6) is trivial given the output of our algorithm.

One popular technique used in tracking is to perform background subtrac-

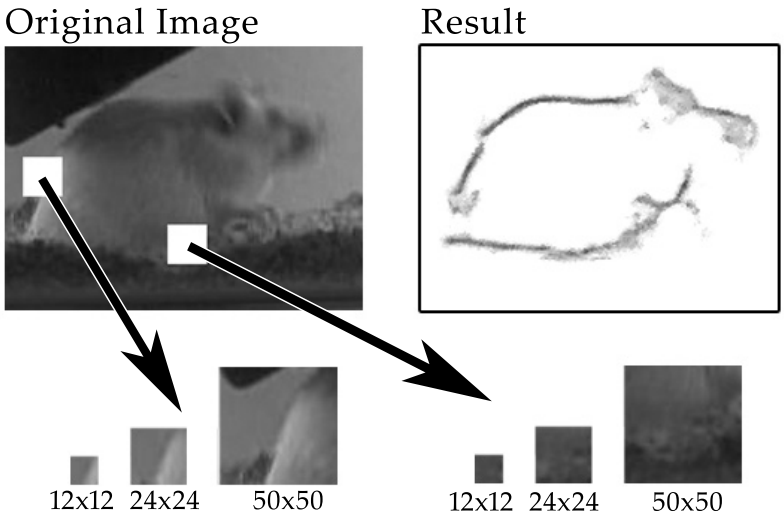


Figure 2.7: Zoomed in view of final testing image for mouse boundary detection and result of classifier. The classifier fails to detect edges near the moving head due to considerable motion blur, an artifact not observed in the training data. Two positive patches of size 24x24 are shown, and corresponding patches of about half and double size. The decision for the left patch can be made based on local information (the small patch is sufficient), but for the right patch context information is crucial (the larger patch is necessary). A patch size of 50x50 is used throughout this chapter to make context information available to the discriminative method.

tion. However, this requires both the camera and the background to be fixed - our framework provides an alternate solution. The authors of [9] use the output of `Pb` as input to their mouse tracker on very similar images, we would expect that our method which is more accurate and faster would prove more useful.

2.4.3 Road detection

Our framework can also be applied to detect roads in satellite images. We obtained the satellite images and corresponding road maps using Google Maps. Each satellite image is aligned with its road map; to generate the ground truth we converted the road map to a binary mask (the process could easily be automated). Road detection is a well known problem, see for example [28]; here we show that our algorithm is directly applicable.

The goal is to classify each pixel as belonging to a road or not. That is we define $S_W(i,j) = 1$ if a location in an image is part of a road according to scene interpretation W, and otherwise 0. In this case, there is only one interpretation W for each image, derived from the corresponding road map, so $\hat{p}(S(i,j)|\mathbf{I}) = S_W(i,j)$. We smooth S_W to allow for some positional uncertainty (alignment of the road to the map is not perfect).

Note that roads are multiple pixels thick. Road detection is not edge detection, rather, the task is pixel assignment. Some results are shown in Figure (2.8).

2.4.4 Edge detection in natural images

One of the strengths of our algorithm is that it can learn an edge detector tuned for a specific domain. If labeled data is available, and the regions or boundaries of interest are of a specific form (for example the boundaries on a mouse or roads in satellite images), then our method outperforms other edge detection algorithms which are designed to respond to all edges, not just specific edges, and cannot take advantage of the particular properties of the edges of interest. However, it is interesting to ask if our algorithm were trained to respond to edges in natural images, how would its performance as a general edge detection algorithm rank?

To train our algorithm to respond to edges in natural images, and to test its performance, we used the Berkeley Segmentation Dataset and Benchmark [48]. We

Figure 2.8: Some results on road detection. Three satellite images were obtained from Google Maps, along with their corresponding road maps. The first two images were used for training, their corresponding road maps were converted to probability distributions of pixel membership. Results on the third image are shown. Close inspection reveals that 'Winchester Dr.' was not detected, it appears that it is much darker than any road in the training data.

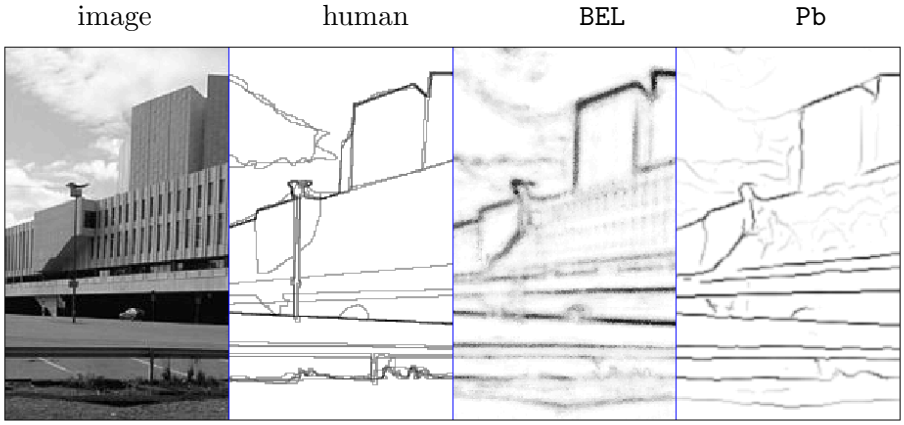


Figure 2.9: Zoomed in view of a test image form the natural image dataset. Qualitative properties of our output can be seen: (1) BEL gives a true probability, not just a measure of edge strength. The strongest responses correlate well with regions where the largest number of people marked edges. (2) The curvature of BEL edges reflects the curvature of human edges, that is straight edges do not become wavy and corners remain sharp (contrast this with the output of Pb). (3) Sharp, clear boundaries (like the building boundaries) give rise to tightly peaked strong responses, boundaries that are harder to localize (near the clouds and grass) give rise to more diffuse weaker responses. This is consistent with human edges near the clouds and grass which also had significant positional uncertainty. The 'F' score for this image is .79 for Pb and .71 for BEL, i.e. according to the benchmark Pb performs significantly better. When computing the overall precision and recall of the output it must be thresholded and thinned, we suspect this adversely affected our measured performance.

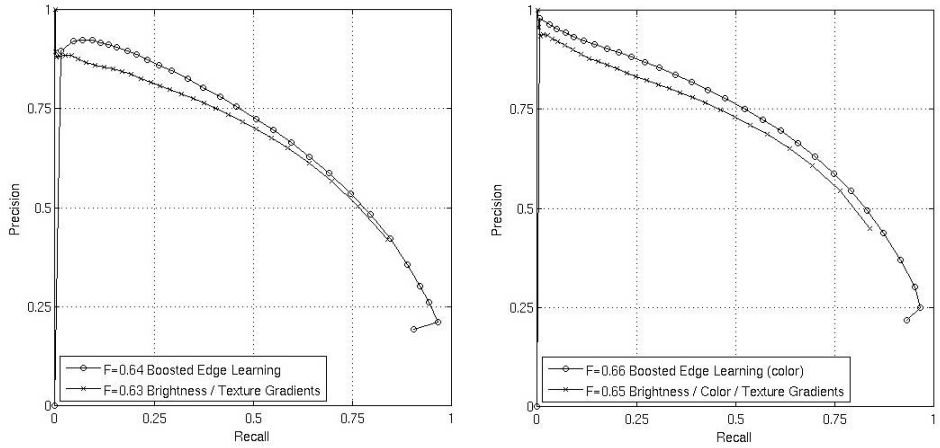


Figure 2.10: Precision recall curves of `BEL` and `Pb` on gray (left) scale and color versions (right) of the Berkeley test set. The overall performance on gray scale images of `BEL` improves on the performance of `Pb`, whose performance is the highest reported in the literature. Similarly for color images, `BEL-Color` outperforms `Pb-Color` (the 'F' score is an overall measure of performance based on all precision/recall values). Training a cascade classifier `BEL-Cascade` for gray scale images failed to give as good of results (not shown), showing the importance of the classification algorithm in our framework. (Curves were obtained from authors of `Pb`).

trained on a subset of the 200 training images ($\frac{1}{6}$ of each of the first 100 images) and applied the algorithm to 100 test images. We repeated the experiment in both gray scale and color (in our framework, adding features based on color simply involves computing the same features we used for the gray scale image over each of the color channels).

Some results on gray scale images are shown in Figure (2.11). Our output gives a true probability whereas the output of `Pb`, like of most edge detection algorithms, gives a strength or confidence of an edge being present, and not a true probability. See Figure (2.9) for a qualitative comparison of the output.

We report quantitative results of our algorithm, calculated using the Berkeley benchmark, in Figure (2.10). More information on how the precision and recall are computed and the significance of the curve can be found in [48]. We compare our performance to multiple variants of `Pb`. The color and gray scale versions of `Pb` have the highest reported performance in the literature. The overall performance of our

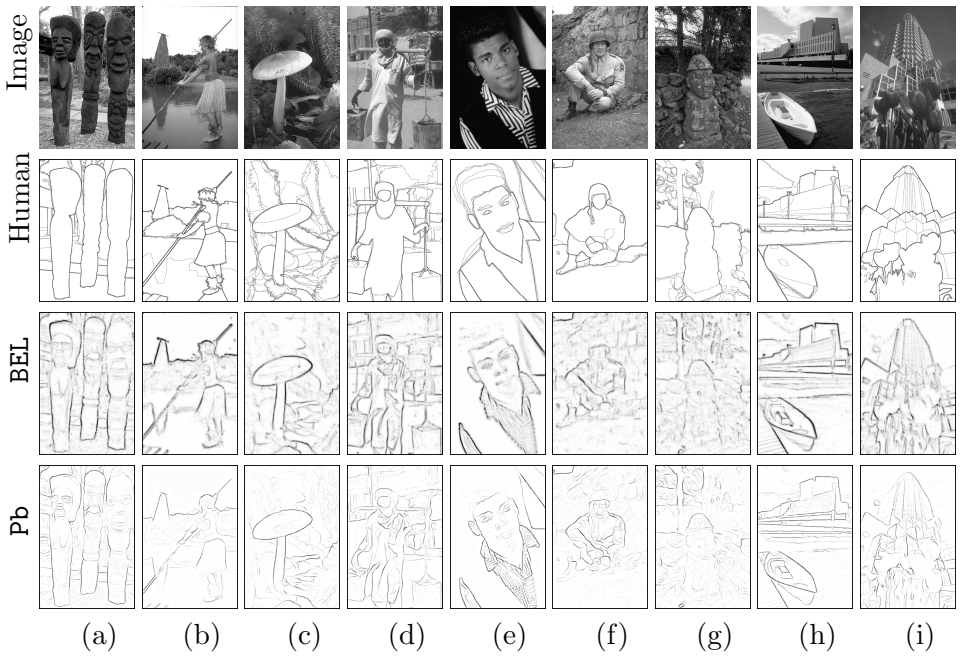


Figure 2.11: The first row contains gray scale images from the Berkeley dataset (http://www.cs.berkeley.edu/projects/vision/grouping/segbench), and the second the overlayed manual segmentations. For each image we give our results and the result of the Berkeley gray scale edge detector for comparison. The images chosen are the same ones as in Figure 15 in [48].

methods on gray scale and color improves upon the performance of the corresponding versions of `Pb`. The performance of all the algorithms shown in Figure (2.10) is significantly higher than the performance of methods based only on brightness gradients, such as the Canny detector. For a full comparison see [48].

We believe that if changed how we do edge thinning, increased the amount of training data we use, trained a deeper tree, or tweaked any number of other factors we could further improve the overall performance, however, this is not central to our agenda, since as mentioned we believe that the true strength of our method lies in its adaptability.

2.5 Discussion

In this chapter, we have presented a learning based algorithm for edge detection which implicitly combines low-level, mid-level and context information across different scales to learn and compute discriminative models for complex patterns. We treat edge detection as a machine learning problem with a very challenging dataset. We use generic, fast features and make no assumptions about what defines an edge, thus avoiding the need to specifically define ad-hoc rules. We use almost identical parameters for training and there are no parameters to specify once the model is learned.

The resulting algorithm is highly adaptive and scalable. We apply it to a number of different datasets, achieving good results. There are a number of applications, such as object detection and tracking, where edge detection could be used but typically is not because edge detectors tend to be inaccurate and give strong responses in uninteresting regions and weak responses in relevant regions. For many of these areas of application sample labeled data can be made available, and we hope that by making the algorithm adaptive it will find use in these applications.

We conclude by acknowledging that there is a limit to how far a discriminative model such as ours can go. The method tends to do very well when adapted to a specific domain, but still has problems on the more general problem of edge detection in natural images, as demonstrated by the difficult images in the Berkeley dataset. We have presented our method in view of an underlying generative model, and argued its merit on the basis of its efficiency and good overall performance. Eventually, however, models that explicitly represent and make use of high-level knowledge must be engaged.

Portions of this chapter are based on "Supervised Learning of Edges and Object Boundaries" by P. Dollár, Z. Tu and S. Belongie [17]. The dissertation author was responsible for implementing the algorithm, performing the experiments and writing the paper.

3

Learning to Traverse Image Manifolds

A number of techniques have been developed for dealing with high dimensional data sets that fall on or near a smooth low dimensional nonlinear manifold. Such data sets arise whenever the number of modes of variability of the data are much fewer than the dimension of the input space, as is the case for image sequences. Unsupervised manifold learning refers to the problem of recovering the structure of a manifold from a set of unordered sample points. Manifold learning is often equated with dimensionality reduction, where the goal is to find an embedding or 'unrolling' of the manifold into a lower dimensional space such that certain relationships between points are preserved. Such embeddings are typically used for visualization, with the projected dimension being 2 or 3.

Image manifolds have also been studied in the context of measuring distance between images undergoing known transformations. For example, the tangent distance [64, 65] between two images is computed by generating local approximations of a manifold from known transformations and then computing the distance between these approximated manifolds. In this chapter, we seek to frame the problem of recovering the structure of a manifold as that of directly learning the transformations a point on a manifold may undergo. Our approach, *Locally Smooth Manifold Learning* (LSML), attempts to learn a warping function $\mathcal{W}$ with d degrees of freedom that can take any point on the manifold and generate its neighbors. LSML recovers a

first order approximation of $\mathcal{W}$, and by making smoothness assumptions on $\mathcal{W}$ can generalize to unseen points.

We show that LSML can recover the structure of the manifold where data is given, and also in regions where it is not, including regions beyond the support of the original data. We propose a number of uses for the recovered warping function $\mathcal{W}$, including embedding with a natural out-of-sample extension, and in the image domain discuss how it can be used for tasks such as computation of tangent distance, image sequence interpolation, compression, and motion transfer. We also show examples where LSML is used to simultaneously learn the structure of multiple "parallel" manifolds, and even generalize to data on new manifolds. Finally, we show that by exploiting the manifold smoothness, LSML is robust under conditions where many embedding methods have difficulty.

Related work is presented in Section 3.1 and the algorithm in Section 3.2. Experiments on point sets and results on images are shown in Sections 3.3 and 3.4, respectively. We conclude in Section 3.5.

3.1 Related Work

Related work can be divided into two categories. The first is the literature on manifold learning, which serves as the foundation for this chapter. The second is work in computer vision and computer graphics addressing image warping and generative models for image formation.

A number of classic methods exist for recovering the structure of a manifold. Principal component analysis (PCA) tries to find a linear subspace that best captures the variance of the original data. Traditional methods for nonlinear manifolds include self organizing maps, principal curves, and variants of multidimensional scaling (MDS) among others, see [33] for a brief introduction to these techniques. Recently the field has seen a number of interesting developments in nonlinear manifold learning. [62] introduced a kernelized version of (PCA). A number of related embedding methods have also been introduced, representatives include LLE [59], ISOMAP [71], and more recently SDE [77]. Broadly, such methods can be classified as spectral embedding techniques [77]; the embeddings they compute

are based on an eigenvector decomposition of an $n \times n$ matrix that represents geometrical relationships of some form between the original n points. Out-of-sample extensions have been proposed [5]. The goal of embedding methods (to find structure preserving embeddings) differs from the goals of LSML (learn to traverse the manifold).

Four methods that we share inspiration with are [8, 39, 4, 57]. [8] employs a novel charting based technique to achieve increased robustness to noise and decreased probability of pathological behavior *vs.* LLE and ISOMAP; we exploit similar ideas in the construction of LSML but differ in motivation and potential applicability. [4] proposed a method to learn the tangent space of a manifold and demonstrated a preliminary illustration of rotating a small bitmap image by about 1°. Work by [39] is based on the notion of learning a model for class specific variation, the method reduces to computing a linear tangent subspace that models variability of each class. [57] shares one of our goals as it addresses the problem of learning Lie groups, the infinitesimal generators of certain geometric transformations.

In image analysis, the number of dimensions is usually reduced via approaches like PCA [53], epitomic representation [36], or generative models as in the realMOVES system developed by Di Bernardo *et al.* [1]. Sometimes, a precise model of the data, *e.g.* for faces [6] or eyes [52], is used to reduce the complexity of the data. Another common approach is simply to have instances of an object in different conditions: [7] start by estimating feature correspondences between a novel input with unknown pose and lighting and a stored labeled example in order to apply an arbitrary warp between pictures. The applications range from video texture synthesis [61] and facial expression extrapolation [13, 72] to face recognition [19] and video rewrite [10].

3.2 Algorithm

Let D be the dimension of the input space, and assume the data lies on a smooth d-dimensional manifold ($d \ll D$). For simplicity assume that the manifold is diffeomorphic with a subset of $\mathbb{R}^d$, meaning that it can be endowed with a global coordinate system (this requirement can easily be relaxed) and that there exists a

continuous bijective mapping $\mathcal{M}$ that converts coordinates $\mathbf{y} \in \mathbb{R}^d$ to points $\mathbf{x} \in \mathbb{R}^D$ on the manifold. The goal of most dimensionality reduction techniques given a set of data points $\mathbf{x}^i$ is to find an embedding $\mathbf{y}^i = \mathcal{M}^{-1}(\mathbf{x}^i)$ that preserves certain properties of the original data like the distances between all points (classical MDS) or the distances or angles between nearby points (*e.g.* spectral embedding methods).

Instead, we seek to learn a warping function $\mathcal{W}$ that can take a point on the manifold and return any neighboring point on the manifold, capturing all the modes of variation of the data. Let us use $\mathcal{W}(\mathbf{x}, \boldsymbol{\epsilon})$ to denote the warping of $\mathbf{x}$, with $\boldsymbol{\epsilon} \in \mathbb{R}^d$ acting on the degrees of freedom of the warp according to the formula $\mathcal{M}$: $\mathcal{W}(\mathbf{x}, \boldsymbol{\epsilon}) = \mathcal{M}(\mathbf{y} + \boldsymbol{\epsilon})$, where $\mathbf{y} = \mathcal{M}^{-1}(\mathbf{x})$. Taking the first order approximation of the above gives: $\mathcal{W}(\mathbf{x}, \boldsymbol{\epsilon}) \approx \mathbf{x} + \mathcal{H}(\mathbf{x})\boldsymbol{\epsilon}$, where each column $\mathcal{H}_{\cdot k}(\mathbf{x})$ of the matrix $\mathcal{H}(\mathbf{x})$ is the partial derivative of $\mathcal{M}$ w.r.t. $\mathbf{y}_k$: $\mathcal{H}_{\cdot k}(\mathbf{x}) = \partial / \partial \mathbf{y}_k \mathcal{M}(\mathbf{y})$. This approximation is valid given $\boldsymbol{\epsilon}$ small enough, hence we speak of $\mathcal{W}$ being an *infinitesimal* warping function.

We can restate our goal of learning to warp in terms of learning a function $\mathcal{H}_\theta : \mathbb{R}^D \to \mathbb{R}^{D \times d}$ parameterized by a variable θ. Only data points $\mathbf{x}^i$ sampled from one or several manifolds are given. For each $\mathbf{x}^i$, the set $\mathcal{N}^i$ of neighbors is then computed (*e.g.* using variants of nearest neighbor such as kNN or ϵNN), with the constraint that two points can be neighbors only if they come from the same manifold. To proceed, we assume that if $\mathbf{x}^j$ is a neighbor of $\mathbf{x}^i$, there then exists an *unknown* $\boldsymbol{\epsilon}^{ij}$ such that $\mathcal{W}(\mathbf{x}^i, \boldsymbol{\epsilon}^{ij}) = \mathbf{x}^j$ to within a good approximation. Equivalently: $\mathcal{H}_\theta(\mathbf{x}^i)\boldsymbol{\epsilon}^{ij} \approx \mathbf{x}^j - \mathbf{x}^i$. We wish to find the best θ in the squared error sense (the $\boldsymbol{\epsilon}^{ij}$ being additional free parameters that must be optimized over). The expression of the error we need to minimize is therefore:

$$\text{error}_1(\theta) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}^i} \left\| \mathcal{H}_\theta(\mathbf{x}^i)\boldsymbol{\epsilon}^{ij} - (\mathbf{x}^j - \mathbf{x}^i) \right\|_2^2 \tag{3.1}$$

Minimizing the above error function can be interpreted as trying to find a warping function that can transform a point into its neighbors. Note, however, that the warping function has only d degrees of freedom while a point may have many more neighbors. This intuition allows us to rewrite the error in an alternate form. Let Δ^i be the matrix where each column is of the form $(\mathbf{x}^j - \mathbf{x}^i)$ for each neighbor of $\mathbf{x}^i$.

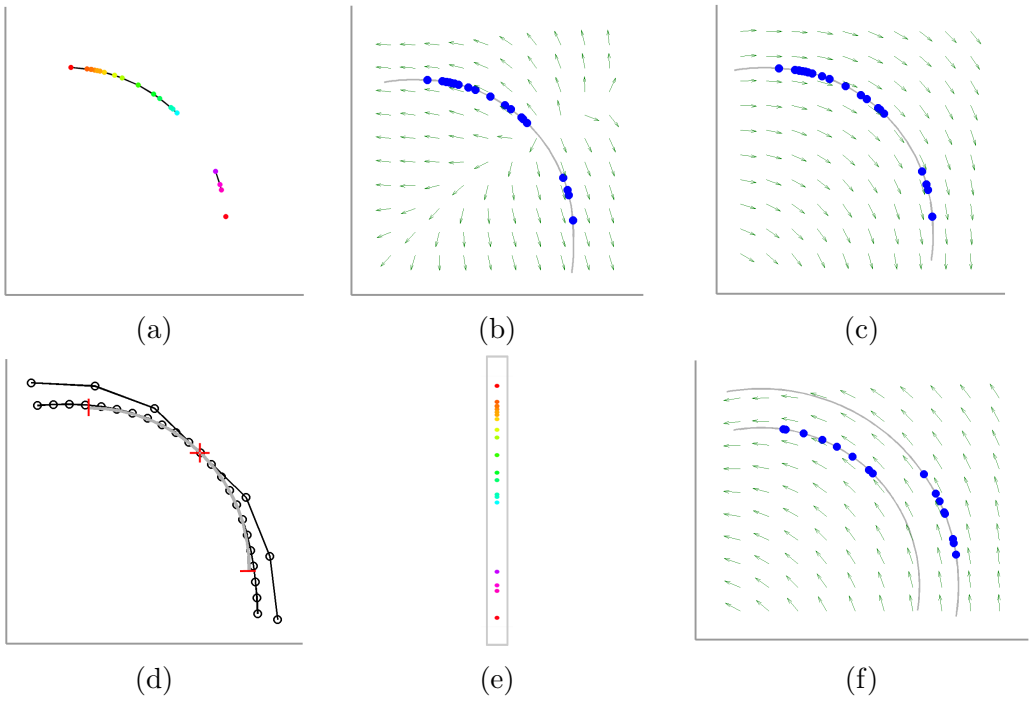


Figure 3.1: **Overview**. Twenty points (n=20) that lie on 1D curve (d=1) in a 2D space (D=2) are shown in (a). Black lines denote neighbors, in this case the neighborhood graph is not connected. We apply LSML to train $\mathcal{H}$ (with $f = 4$ RBFs). $\mathcal{H}$ maps points in $\mathbb{R}^2$ to tangent vectors; in (b) tangent vectors computed over a regularly spaced grid are displayed, with original points (blue) and curve (gray) overlayed. Tangent vectors near original points align with the curve, but note the seam through the middle. Regularization fixes this problem (c), the resulting tangents roughly align to the curve along its entirety. We can traverse the manifold by taking small steps in the direction of the tangent; (d) shows two such paths, generated starting at the red plus and traversing outward in large steps (outer curve) and finer steps (inner curve). This generates a coordinate system for the curve resulting in a 1D embedding shown in (e). In (f) two parallel curves are shown, with n=8 samples each. Training a common $\mathcal{H}$ results in a vector field that more accurately fits each curve than training a separate $\mathcal{H}$ for each (if the structure of the two manifolds was very different this need not be the case).

Let $\Delta^i = U^i \Sigma^i {V^i}^\top$ be the thin singular value decomposition of Δ^i. Then, one can show (see Section 3.6) that error_1 is equivalent to the following:

$$\text{error}_2(\theta) = \min_{\{E^i\}} \sum_{i=1}^{n} \left\| \mathcal{H}_\theta(\mathbf{x}^i) E^i - U^i \Sigma^i \right\|_F^2 \tag{3.2}$$

Here, the matrices E^i are the additional free parameters. Minimizing the above can be interpreted as searching for a warping function that directly explains the modes of variation at each point. This form is convenient since we no longer have to keep track of neighbors. Furthermore, if there is no noise and the linearity assumption holds there are at most d non-zero singular values. In practice we use the truncated SVD, keeping at most $2d$ singular values, allowing for significant computational savings.

We now give the remaining details of LSML for the general case (see Section 3.6). For the case of images, we present an efficient version in Section 3.4 which uses some basic domain knowledge to avoid solving a large regression. Although potentially any regression technique is applicable, a linear model is particularly easy to work with. Let $\mathbf{f}^i$ be f features computed over $\mathbf{x}^i$. We can then define $\mathcal{H}_\theta(\mathbf{x}^i) = [\Theta^1 \mathbf{f}^i \cdots \Theta^D \mathbf{f}^i]^\top$, where each Θ^k is a $d \times f$ matrix. Re-arranging error_2 gives:

$$\text{error}_{lin}(\theta) = \min_{\{E^i\}} \sum_{i=1}^{n} \sum_{k=1}^{D} \left\| {\mathbf{f}^i}^\top {\Theta^k}^\top E^i - U^i_{k\cdot} \Sigma^i \right\|_2^2 \tag{3.3}$$

Solving simultaneously for E and Θ is complex, but if either E or Θ is fixed, solving for the remaining variable becomes a least squares problem (an equation of the form $AXB = C$ can be rewritten as $B^\top \otimes A \cdot \text{vec}(X) = \text{vec}(C)$, where $\otimes$ denotes the Kronecker product and vec the matrix vectorization function). To solve for θ, we use an alternating minimization procedure. In all experiments in this chapter we perform 30 iterations of the above procedure, and while local minima do not seem to be to prevalent, we randomly restart the procedure 5 times. Finally, nowhere in the construction have we enforced that the learned tangent vectors be orthogonal (such a constraint would only be appropriate if the manifold was isometric to a plane).

To avoid numerically unstable solutions we regularize the error:

$$\text{error}'_{lin}(\theta) = \text{error}_{lin}(\theta) + \lambda_E \sum_{i=1}^{n} \left\|E^i\right\|_F^2 + \lambda_\theta \sum_{k=1}^{D} \left\|\Theta^k\right\|_F^2 \tag{3.4}$$

For the features we use radial basis functions (RBFs) [33], the number of basis functions, f, being an additional parameter. Each basis function is of the form $f^j(\mathbf{x}) = \exp(-\|\mathbf{x} - \boldsymbol{\mu}^j\|_2^2/2\sigma^2)$ where the centers $\boldsymbol{\mu}^j$ are obtained using K-means clustering on the original data with f clusters and the width parameter σ is set to be twice the average of the minimum distance between each cluster and its nearest neighbor center. The feature vectors are then simply defined as $\mathbf{f}^i = [f^1(\mathbf{x}^i) \cdots f^p(\mathbf{x}^i)]^\top$. The parameter f controls the smoothness of the final mapping $\mathcal{H}_\theta$; larger values result in mappings that better fit local variations of the data, but whose generalization abilities to other points on the manifold may be weaker. This is exactly analogous to the standard supervised setting and techniques like cross validation could be used to optimize over f.

3.3 Experiments on Point Sets

We begin with a discussion on the intuition behind various aspects of LSML. We then show experiments demonstrating the robustness of the method, followed by a number of applications. In the figures that follow we make use of color/shading to indicate point correspondences, for example when we show the original point set and its embedding.

LSML learns a function $\mathcal{H}$ from points in $\mathbb{R}^D$ to tangent directions that agree, up to a linear combination, with estimated tangent directions at the original training points of the manifold. By constraining $\mathcal{H}$ to be smooth (through use of a limited number of RBFs), we can compute tangents at points not seen during training, including points that may not lie on the underlying manifold. This generalization ability of $\mathcal{H}$ will be central to the types of applications considered. Finally, given multiple non-overlapping manifolds with similar structure, we can train a single $\mathcal{H}$ to correctly predict the tangents of each, allowing information to be shared. Fig. 3.1 gives a visual tutorial of these different concepts.

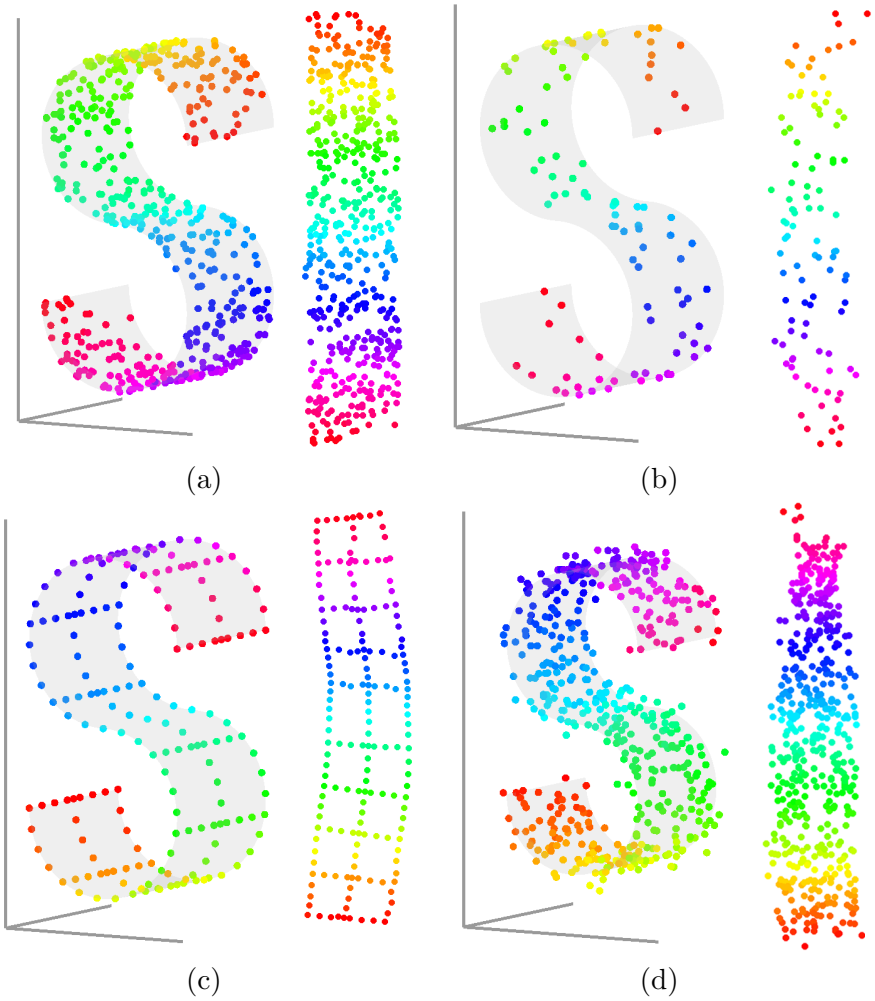


Figure 3.2: **Robustness**. LSML used to recover the embedding of the $\mathcal{S}$-curve under a number of sampling conditions. In each plot we show the original points along with the computed embedding (rotated to align vertically), correspondence is indicated by coloring/shading (color was determined by the y-coordinate of the embedding). In each case LSML was run with $f = 8$, $d = 2$, and neighbors computed by ϵNN with $\epsilon = 1$ (the height of the curve is 4). The embeddings shown were recovered from data that was: (a) densely sampled (n=500) (b) sparsely sampled (n=100), (c) highly structured (n=190), and (d) noisy (n=500, random Gaussian noise with $\sigma = .1$). In each case LSML recovered the correct embedding. For comparison, LLE recovered good embeddings for (a) and (c) and ISOMAP for (a),(b), and (c). The experiments were repeated a number of times yielding similar results. For a discussion see the text.

LSML appears quite robust. Fig. 3.2 shows LSML successfully applied for recovering the embedding of the "$\mathcal{S}$-curve" under a number of sampling conditions (similar results were obtained on the "Swiss-roll"). After $\mathcal{H}$ is learned, the embedding is computed by choosing a random point on the manifold and establishing a coordinate system by traversing outward (the same procedure can be used to embed novel points, providing a natural out-of-sample extension). Here we compare only to LLE and ISOMAP using published code. The densely sampled case, Fig. 3.2(a), is comparatively easy and a number of methods have been shown to successfully recover an embedding. On sparsely sampled data, Fig. 3.2(b), the problem is more challenging; LLE had problems for $n < 250$ (lowering LLE's regularization parameter helped somewhat). Real data need not be uniformly sampled, see Fig. 3.2(c). In the presence of noise Fig. 3.2(d), ISOMAP and LLE performed poorly. A single outlier can distort the shortest path computed by ISOMAP, and LLE does not directly use global information necessary to disambiguate noise. Other methods are known to be robust [8], and in [82] the authors propose a method to "smooth" a manifold as a preprocessing step for manifold learning algorithms; however a full comparison is outside the scope of this chapter.

Having learned $\mathcal{H}$ and computed an embedding, we can also backproject from a point $\mathbf{y} \in \mathbb{R}^d$ to a point $\mathbf{x}$ on the manifold by first finding the coordinate of the closest point $\mathbf{y}^i$ in the original data, then traversing from $\mathbf{x}^i$ by $\epsilon_j = \mathbf{y}_j - \mathbf{y}^i_j$ along each tangent direction j (see Fig. 3.1(d)). Fig. 3.3(a) shows tangents and an embedding recovered by LSML on the Swiss-roll. In Fig. 3.3(b) we backproject from a grid of points in $\mathbb{R}^2$; by linking adjacent sets of points to form quadrilaterals we can display the resulting backprojected points as a surface. In Fig. 3.3(c), we likewise do a backprojection (this time keeping all the original points), however we backproject grid points well below and above the support of the original data. Although there is no ground truth here, the resulting extension of the surface seems "natural". Fig. 3.3(d) shows the reconstruction of a unit hemisphere by traversing outward from the topmost point. There is no isometric mapping (preserving distance) between a hemisphere and a plane, and given a sphere there is actually not even a conformal mapping (preserving angles). In the latter case an embedding is not possible, however, we can still easily recover $\mathcal{H}$ for both (only hemisphere results are shown).

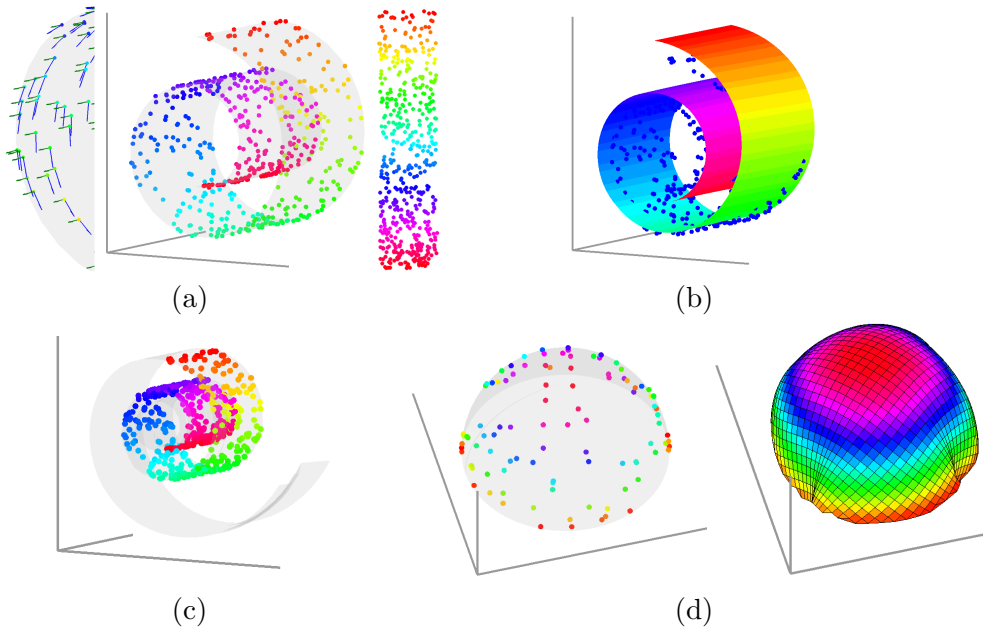


Figure 3.3: **Reconstruction**. Reconstruction examples are used to demonstrate quality and generalization of $\mathcal{H}$. (a) Points sampled from the Swiss-roll manifold (middle), some recovered tangent vectors in a zoomed-in region (left) and embedding found by LSML (right). Here $n = 500$ $f = 20$, $d = 2$, and neighbors were computed by ϵNN with $\epsilon = 4$ (height of roll is 20). Reconstruction of Swiss-roll (b), created by a backprojection from regularly spaced grid points in the embedding (traversal was done from a single original point located at the base of the roll, see text for details). Another reconstruction (c), this time using all points and extending the grid well beyond the support of the original data. The Swiss-roll is extended in a reasonable manner both inward (occluded) and outward. (d) Reconstruction of unit hemisphere (LSML trained with $n = 100$ $f = 6$, $d = 2$, ϵNN with $\epsilon = .3$) by traversing outward from topmost point, note reconstruction in regions with no points.

3.4 Results on Images

Before continuing, we consider potential applications of $\mathcal{H}$ in the image domain, including tangent distance estimation, nonlinear interpolation, extrapolation, compression, and motion transfer. We refer to results on point-sets to aid visualization. Tangent distance estimation: $\mathcal{H}$ computes the tangent and can be used directly in invariant recognition schemes such as [65]. Compression: Fig. 3.3(b,d) suggest how given a reference point and $\mathcal{H}$ nearby points can be reconstructed using d numbers (with distortion increasing with distance). Nonlinear interpolation and extrapolation: points can be generated within and beyond the support of given data (*cf*. Fig. 3.3); of potential use in tasks such as frame rate up-conversion, reconstructing dropped frames and view synthesis. Motion transfer: for certain classes of manifolds with "parallel" structure (*cf*. Fig. 3.1(f)), a recovered warp may be used on an entirely novel image. These applications will depend not only on the accuracy of the learned $\mathcal{H}$ but also on how close a set of images is to a smooth manifold.

The key insight to working with images is that although images can live in very high dimensional spaces (with $D \approx 10^6$ quite common), we do not have to learn a transformation with that many parameters. Let $\mathbf{x}$ be an image and $\mathcal{H}_{.k}(\mathbf{x})$, $k \in [1, d]$, be the d tangent images. Here we assume that each pixel in $\mathcal{H}_{.k}(\mathbf{x})$ can be computed based only on the information in $s \times s$ patch centered on the corresponding pixel in $\mathbf{x}$. Thus, instead of learning a function $\mathbb{R}^D \rightarrow \mathbb{R}^{D \times d}$ we learn a function $\mathbb{R}^{s^2} \rightarrow \mathbb{R}^d$, and to compute $\mathcal{H}$ we apply the per patch function at each of the D locations in the image. The resulting technique scales independently of D, in fact different sized images can be used. The per patch assumption is not always suitable, most notably for transformations that are based only on image coordinate and are independent of appearance.

The approach of Section 3.2 needs to be slightly modified to accommodate patches. We rewrite each image $\mathbf{x}^i \in \mathbb{R}^D$ as a $s^2 \times D$ matrix X^i where each row contains pixels from one patch in $\mathbf{x}^i$ (in training we sub-sample patches). Patches from all the images are clustered to obtain the f RBFs; each X^i is then transformed to a $f \times D$ matrix F^i that contains the features computed for each patch. The per patch linear model can now be written as $\mathcal{H}_\theta(\mathbf{x}^i) = (\Theta F^i)^\top$, where Θ is a $d \times f$

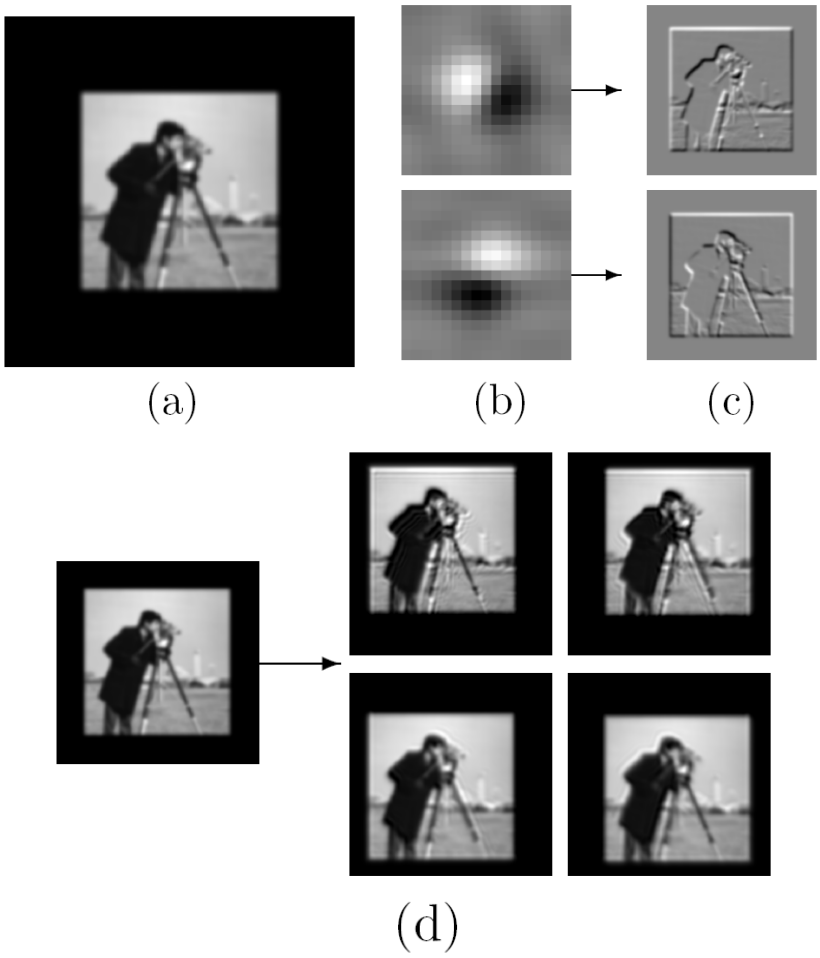


Figure 3.4: **The translation manifold**. Here $F^i = X^i$; $s = 17$, $d = 2$ and 9 sets of 6 translated images each were used (not including the cameraman). (a) Zero padded, smoothed test image $\mathbf{x}$. (b) Visualization of learned Θ, see text for details. (c) $\mathcal{H}_\theta(\mathbf{x})$ computed via convolution. (d) Several transformations obtained after multiple steps along manifold for different linear combinations of $\mathcal{H}_\theta(\mathbf{x})$. Some artifacts due to error propagation start to appear in the top figures.

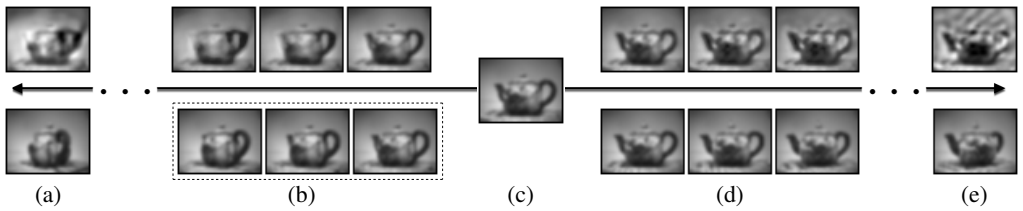


Figure 3.5: **Manifold generated by out-of-plane rotation of a teapot** (data from [77], sub-sampled and smoothed). Here, $d = 1$, $f = 400$ and roughly 3000 patches of width $s = 13$ were sampled from 30 frames. Bottom row shows the ground truth images; dashed box contains 3 of 30 training images, representing $\sim 8^\circ$ of physical rotation. The top row shows the learned transformation applied to the central image. By observing the tip, handle and the two white blobs on the teapot, and comparing to ground truth data, we can observe the quality of the learned transformation on seen data (b) and unseen data (d), both starting from a single frame (c). The outmost figures (a)(e) shows failure for large rotations.

matrix (compare with the D Θs needed without the patch assumption). The error function, which is minimized in a similar way (see Section 3.6), becomes:

$$\text{error}_{img}(\Theta) = \min_{\{E^i\}} \sum_{i=1}^{n} \left\| {F^i}^\top \Theta^\top E^i - U^i \Sigma^i \right\|_F^2 \tag{3.5}$$

We begin with the illustrative example of translation (Fig. 3.4). Here, RBFs were not used, instead $F^i = X^i$. The learned Θ is a $2 \times s^2$ matrix, which can be visualized as two $s \times s$ images as in Fig. 3.4(b). These resemble derivative of Gaussian filters, which are in fact the infinitesimal generates for translation [57]. Computing the dot product of each column of Θ with each patch can be done using a convolution. Fig. 3.4 shows applications of the learned transformations, which resemble translations with some artifacts.

Fig. 3.5 shows the application of LSML for learning out-of-plane rotation of a teapot. On this size problem training LSML (in MATLAB) takes a few minutes; convergence occurs within about 10 iterations of the minimization procedure. $\mathcal{H}_\theta(\mathbf{x})$ for novel $\mathbf{x}$ can be computed with f convolutions (to compute cross correlation) and is also fast. The outer frames in Fig. 3.5 highlight a limitation of the approach: with every successive step error is introduced; eventually significant error can accumulate.

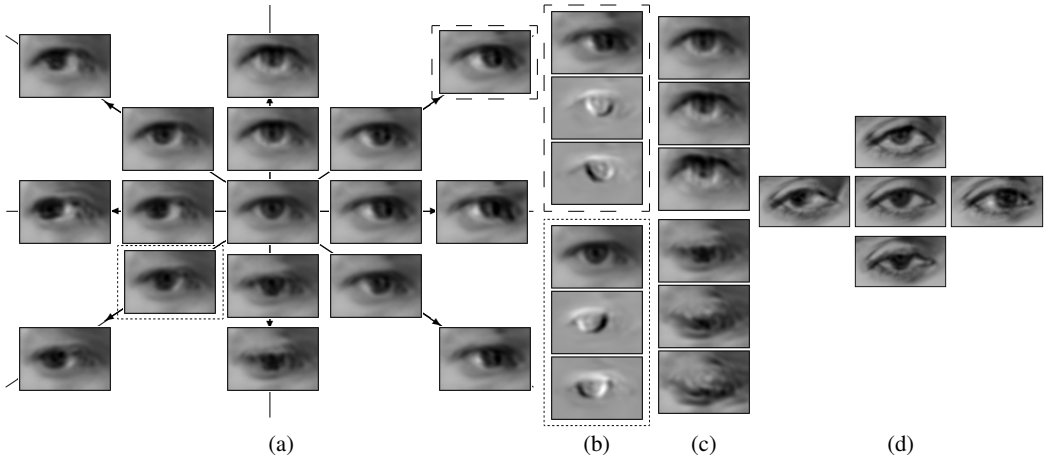


Figure 3.6: **Traversing the eye manifold**. LSML trained on one eye moving along five different lines (3 vertical and 2 horizontal). Here $d = 2$, $f = 600$, $s = 19$ and around 5000 patches were sampled; 2 frames were considered neighbors if they were adjacent in time. Figure (a) shows images generated from the central image. The inner 8 frames lie just outside the support of the training data (not shown), the outer 8 are extrapolated beyond its support. Figure (b) details $\mathcal{H}_\theta(\mathbf{x})$ for two images in a warping sequence: a linear combination can lead the iris/eyelid to move in different directions (*e.g.* the sum would make the iris go up). Figure (c) shows extrapolation far beyond the training data, *i.e.* an eye wide open and fully closed. Finally, Figure(d) shows how the eye manifold we learned on one eye can be applied on a novel eye not seen during training.

Here, we used a step size which gives roughly 10 interpolated frames between each pair of original frames. With out-of-plane rotation, information must be created and the problem becomes ambiguous (multiple manifolds can intersect at a single point), hence generalization across images is not expected to be good.

In Fig. 3.6, results are shown on an eye manifold with 2 degrees of freedom. LSML was trained on sparse data from video of a single eye; $\mathcal{H}_\theta$ was used to synthesize views within and also well outside the support of the original data (*cf*. Fig. 3.6(c)). In Fig. 3.6(d), we applied the transformation learned from one person's eye to a single image of another person's eye (taken under the same imaging conditions). LSML was able to start from the novel test image and generate a convincing series of transformations. Thus, motion transfer was possible - $\mathcal{H}_\theta$ trained on one series of images generalized to a different set of images.

3.5 Discussion

In this chapter we presented an algorithm, Locally Smooth Manifold Learning, for learning the structure of a manifold. Rather than pose manifold learning as the problem of recovering an embedding, we posed the problem in terms of learning a warping function for traversing the manifold. Smoothness assumptions on $\mathcal{W}$ allowed us to generalize to unseen data. Proposed uses of LSML include tangent distance estimation, frame rate up-conversion, video compression and motion transfer.

Future work includes scaling the implementation to handle large datasets; the goal would be to integrate LSML into recognition systems to provide increased invariance to transformations.

This chapter is in part based on the paper "Learning to Traverse Image Manifolds" by P. Dollár, V. Rabaud and S. Belongie [15]. The dissertation author proposed the initial idea, developed much of the mathematics and code, and wrote the bulk of the paper.

3.6 Mathematical Details

Notation:

a	Scalar
$\mathbf{a}$	Vector
$\mathbf{a}^i$	Vector indexed for some purpose
a_i	i^{th} element of the vector $\mathbf{a}$
A	Matrix
A^i	Matrix indexed for some purpose
$A_{\cdot j}$	j^{th} column of the matrix A
$A_{i\cdot}$	i^{th} row of the matrix A
$\text{vec}\,(A)$	vector version of the matrix A
A^+	Pseudo-inverse of A
$\|\mathbf{x}\|_2^2$	squared L^2 norm of $\mathbf{x}$
$\|A\|_F^2$	squared Frobenius norm of A

Variables:

D		dim. of original space
d		dim. of projected space
n		number of data points
f		number features per point
$\mathbf{x}^i$	$[D \times 1]$	$i \in [n]$, data point
$\mathbf{f}^i$	$[f \times 1]$	$i \in [n]$, features of $\mathbf{x}^i$
$\mathcal{N}_i$		indices of neighbors of $\mathbf{x}^i$
$\mathcal{H}_\theta$		$\mathcal{H}_\theta : \mathbb{R}^D \to \mathbb{R}^{D \times d}$
H^i	$[D \times d]$	$H^i = \mathcal{H}_\theta(\mathbf{x}^i)(\theta \text{ fixed})$
$\boldsymbol{\epsilon}^{ij}, E^i$		free parameters
Δ^i	$[D \times \lvert\mathcal{N}_i\rvert]$	$\Delta^i_{\cdot j} = \mathbf{x}^j - \mathbf{x}^i$
$U^i \Sigma^i {V^i}^\top$		SVD of Δ^i

3.6.1 Equivalence of $\text{error}_1(\theta)$ and $\text{error}_2(\theta)$

$$\text{error}_1(\theta) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}^i} \left\| \mathcal{H}_\theta(\mathbf{x}^i)\boldsymbol{\epsilon}^{ij} - (\mathbf{x}^j - \mathbf{x}^i) \right\|_2^2$$

$$\text{error}_2(\theta) = \min_{\{E^i\}} \sum_{i=1}^{n} \left\| \mathcal{H}_\theta(\mathbf{x}^i)E^i - U^i\Sigma^i \right\|_F^2$$

Fixing θ we can solve for each $\boldsymbol{\epsilon}^{ij}$ and E^i and rewrite $\text{error}_1(\theta)$ and $\text{error}_2(\theta)$ as:

$$\text{error}_1(\theta) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}^i} \left\| H^i\boldsymbol{\epsilon}^{ij} - \Delta^i_{\cdot j} \right\|_2^2 = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}^i} \left\| \left(H^i H^{i^+} - I \right) \Delta^i_{\cdot j} \right\|_2^2$$

$$\text{error}_2(\theta) = \min_{\{E^i\}} \sum_{i=1}^{n} \left\| H^i E^i - U^i\Sigma^i \right\|_F^2 = \sum_{i=1}^{n} \left\| \left(H^i H^{i^+} - I \right) U^i\Sigma^i \right\|_F^2$$

We show the equivalence of these two forms, using the facts that $\|A\|_F^2 = tr(A \cdot A^\top)$, $\|\mathbf{x}\|_2^2 = tr(\mathbf{x}\mathbf{x}^\top)$, and for any unitary matrix V the following holds: $\sum_j \left({V_{j\cdot}}^\top \cdot V_{j\cdot} \right) = I$. Proof:

$$\begin{aligned}
\text{error}_1(\theta) &= \sum_{i=1}^{n} \sum_{j \in \mathcal{N}^i} \left\| \left(H^i H^{i^+} - I \right) \Delta^i_{\cdot j} \right\|_2^2 \\
&= \sum_i \sum_{j \in \mathcal{N}_i} tr\left(\left(H^i H^{i^+} - I \right) \Delta^i_{\cdot j} \Delta^i_{\cdot j} \top \left(H^i H^{i^+} - I \right)^\top \right) \\
&= \sum_i \sum_{j \in \mathcal{N}_i} tr\left(\left(H^i H^{i^+} - I \right) \cdot U^i\Sigma^i {V^i_{\cdot j}}^\top \cdot V^i_{\cdot j} {\Sigma^i}^\top {U^i}^\top \cdot \left(H^i H^{i^+} - I \right)^\top \right) \\
&= \sum_i tr\left(\left(H^i H^{i^+} - I \right) \cdot U^i\Sigma^i \cdot \sum_{j \in \mathcal{N}_i} \left({V^i_{\cdot j}}^\top \cdot V^i_{\cdot j} \right) \cdot {\Sigma^i}^\top {U^i}^\top \cdot \left(H^i H^{i^+} - I \right)^\top \right) \\
&= \sum_i tr\left(\left(H^i H^{i^+} - I \right) \cdot U^i\Sigma^i \cdot {\Sigma^i}^\top {U^i}^\top \cdot \left(H^i H^{i^+} - I \right)^\top \right) \text{ as } V^i \text{ is unitary} \\
&= \sum_i \left\| \left(H^i H^{i^+} - I \right) U^i\Sigma^i \right\|_F^2 = \text{error}_2(\theta)
\end{aligned}$$

3.6.2 Minimization for General Manifolds

Here we use the linear parametrization of $\mathcal{H}_\Theta$: $\mathcal{H}_\Theta$ is parameterized by $\Theta = (\Theta^1, \cdots, \Theta^D)$, where each Θ^k is a $d \times f$ matrix. The total number of parameters is thus Ddf. $\mathcal{H}_\Theta$ has the following form:

$$\mathcal{H}_\Theta(\mathbf{x}^i) = \left[\Theta^1 \mathbf{f}^i \cdots \Theta^D \mathbf{f}^i\right]^\top$$

Plugging $\mathcal{H}_\Theta$ into $\text{error}_2(\theta)$ gives the new error function $\text{error}_{lin}(\theta)$:

$$\begin{aligned}
\text{error}_{lin}(\theta) &= \min_{\{E^i\}} \sum_{i=1}^{n} \left\| \mathcal{H}_\Theta(\mathbf{x}^i) E^i - U^i \Sigma^i \right\|_F^2 \\
&= \min_{\{E^i\}} \sum_{i=1}^{n} \sum_{k=1}^{D} \left\| \mathbf{f}^{i\top} \Theta^{k\top} E^i - U^i_{k\cdot} \Sigma^i \right\|_2^2
\end{aligned}$$

Minimize with respect to Θ:

1. Initialize Θ randomly.
2. Loop:
   (a) For each i, solve for the best E^i given the Θ^k's:

$$\begin{aligned}
E^i &= \arg\min_{\{E^i\}} \sum_{k=1}^{D} \left\| \mathbf{f}^{i\top} \Theta^{k\top} E^i - U^i_{k\cdot} \Sigma^i \right\|_2^2 \\
&= \arg\min_{\{E^i\}} \left\| H^i E^i - U^i \Sigma^i \right\|_F^2 \\
&= H^{i+} U^i \Sigma^i
\end{aligned}$$

(b) For each k, solve for the best Θ^k given the E^i's:

$$\begin{aligned}\Theta^k &= \arg\min_{\Theta^k} \sum_{i=1}^{n} \left\| \mathbf{f}^{i\top} \Theta^{k\top} E^i - U^i_{k\cdot} \Sigma^i \right\|_2^2 \\ &= \arg\min_{\Theta^k} \sum_{i=1}^{n} \left\| \left(E^{i\top} \otimes \mathbf{f}^{i\top} \right) \mathrm{vec}\left(\Theta^{k\top} \right) - \mathrm{vec}\left(U^i_{k\cdot} \Sigma^i \right) \right\|_2^2\end{aligned}$$

Since $\mathrm{vec}\left(U^i_{k\cdot}\Sigma^i\right) = \Sigma^i {U^i_{k\cdot}}^\top$, the least squares solution for Θ^k becomes:

$$\mathrm{vec}\left(\Theta^{k\top}\right) = \begin{bmatrix} E^{1\top} \otimes \mathbf{f}^{1\top} \\ \vdots \\ E^{n\top} \otimes \mathbf{f}^{n\top} \end{bmatrix}^+ \begin{bmatrix} \Sigma^1 {U^1_{k\cdot}}^\top \\ \vdots \\ \Sigma^n {U^n_{k\cdot}}^\top \end{bmatrix}$$

Adding a regularization term on the E^is and Θ^ks is important for achieving good solutions. The error function becomes:

$$\mathrm{error}'_{lin}(\theta) = \mathrm{error}_{lin}(\theta) + \lambda_E \sum_{i=1}^{n} \left\| E^i \right\|_F^2 + \lambda_\theta \sum_{k=1}^{D} \left\| \Theta^k \right\|_F^2$$

When computing the optimal Θ^k and E^i, the above equations needs to be modified only slightly. The scheme is similar to solving for $\mathbf{x} = \arg\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{x}\|_2^2$, in which case the solution is simply obtained by taking the derivative and setting to 0: $\mathbf{x} = (A^\top A + \lambda I)^{-1} A^\top \mathbf{b}$. The above equations are adjusted similarly.

3.6.3 Minimization for Image Manifolds

$K \leq D$ patches of size $s \times s$ are extracted from each image. After clustering the patches to obtain f clusters, for each patch a length f feature vector is computed, and for each image i the feature vectors for each patch are stacked in a matrix F^i (of size $f \times K$). For images, there is a single Θ (of size $d \times f$). The error, again derived from $\text{error}_2(\theta)$ is:

$$\begin{aligned}\text{error}_{img}(\theta) &= \min_{\{E^i\}} \sum_{i=1}^{n} \left\| \mathcal{H}_\Theta(\mathbf{x}^i) E^i - U^i \Sigma^i \right\|_F^2 \\ &= \min_{\{E^i\}} \sum_{i=1}^{n} \left\| {F^i}^\top \Theta^\top E^i - U^i \Sigma^i \right\|_F^2\end{aligned}$$

Minimize with respect to Θ:

1. Initialize Θ randomly.
2. Loop:
   (a) For each i, solve for the best E^i given Θ:

$$E^i = \left({F^i}^\top \Theta^\top \right)^+ U^i \Sigma^i$$

   (b) Solve for the best Θ given the E^i's:

$$\begin{aligned}\Theta &= \arg\min_{\Theta} \sum_{i=1}^{n} \left\| {F^i}^\top \Theta^\top E^i - U^i \Sigma^i \right\|_F^2 \\ &= \arg\min_{\Theta} \sum_{i=1}^{n} \left\| \left({E^i}^\top \otimes {F^i}^\top \right) \text{vec}\left(\Theta^\top \right) - \text{vec}\left(U^i \Sigma^i \right) \right\|_F^2\end{aligned}$$

Hence, $\text{vec}\left(\Theta^\top\right)$ is the least square solution of:

$$\begin{bmatrix} A^{11} \\ \vdots \\ A^{nD} \end{bmatrix} \text{vec}\left(\Theta^\top \right) = \begin{bmatrix} \text{vec}\left(U^1 \Sigma^1 \right) \\ \vdots \\ \text{vec}\left(U^n \Sigma^n \right) \end{bmatrix} = \begin{bmatrix} U^1_{\cdot 1} \Sigma^1_{11} \\ \vdots \\ U^n_{\cdot D} \Sigma^n_{DD} \end{bmatrix}$$

where $A^{ij} = {E^i_{\cdot j}}^\top \otimes {F^i}^\top$. As the left matrix is of size $nDK \times df$, we do not solve this least squares problem directly, but rather transform the equation to:

$$\begin{bmatrix} A^{11} \\ \vdots \\ A^{nD} \end{bmatrix}^\top \begin{bmatrix} A^{11} \\ \vdots \\ A^{nD} \end{bmatrix} \operatorname{vec}\left(\Theta^\top\right) = \begin{bmatrix} A^{11} \\ \vdots \\ A^{nD} \end{bmatrix}^\top \begin{bmatrix} U^1_{\cdot 1}\Sigma^1_{11} \\ \vdots \\ U^n_{\cdot D}\Sigma^n_{DD} \end{bmatrix}$$

$$\left(\sum_{i=1}^{n}\sum_{j=1}^{D} {A^{ij}}^\top A^{ij}\right) \operatorname{vec}\left(\Theta^\top\right) = \sum_{i=1}^{n}\sum_{j=1}^{D} \Sigma^i_{jj} {A^{ij}}^\top U^i_{\cdot j}$$

This equation is simpler to solve as the left matrix is only $df \times df$ and typically full rank.

Adding a regularization term is similar to the case for general manifolds.

3.6.4 lsml for General Manifolds

INPUT

$\mathbf{x}^i$:	input data points ($1 \leq i \leq n$) in $\mathbb{R}^D$
$\mathcal{N}^i$:	neighbors of $\mathbf{x}^i$ (*e.g.* using kNN)
d:	manifold dimensionality
f:	number of RBFs (controls smoothness)

PRECOMPUTATIONS

$\boldsymbol{\mu}^j, \sigma$:	Run K-means on the $\mathbf{x}^i$ with f clusters; set $\boldsymbol{\mu}^j$ to the cluster centers. Set σ to twice the average distance between a cluster and its nearest neighbor.
$\mathbf{f}^i$:	Features: $\mathbf{f}^i_j = \exp(\left\|\mathbf{x}^i - \boldsymbol{\mu}^j\right\|_2^2 / 2\sigma^2)$
U^i, Σ^i:	For each $\mathbf{x}_i$, compute the difference to neighbor matrix Δ^i and apply the SVD to get: $\Delta^i = U^i \Sigma^i {V^i}^\top$

MINIMIZATION

$\forall k$, initialize Θ^k to a random $d \times f$ matrix.

while $\text{error}_{lin}(\theta)$ still decreases **do**

$\forall i$, solve for the best E^i given the Θ^ks:

$$H^i = \left[\Theta^1 \mathbf{f}^i \cdots \Theta^D \mathbf{f}^i\right]^\top$$

$$E^i = {H^i}^+ U^i \Sigma^i$$

$\forall k$, solve for the best Θ^k given the E^i's:

$$\text{vec}\left({\Theta^k}^\top\right) = \begin{bmatrix} {E^1}^\top \otimes {\mathbf{f}^1}^\top \\ \vdots \\ {E^n}^\top \otimes {\mathbf{f}^n}^\top \end{bmatrix}^+ \begin{bmatrix} {\Sigma^1 U^1_{k\cdot}}^\top \\ \vdots \\ {\Sigma^n U^n_{k\cdot}}^\top \end{bmatrix}$$

3.6.5 lsml for Image Manifolds

INPUT

$\mathbf{x}^i$:	input images ($1 \leq i \leq n$) in $\mathbb{R}^D$
$\mathcal{N}^i$:	neighbors of $\mathbf{x}^i$ (*e.g.* from proximity in video)
d:	manifold dimensionality
f:	number of RBFs (controls smoothness)
s:	patch width/height

PRECOMPUTATIONS

$\boldsymbol{\mu}^j, \sigma$:	Run K-means on large number of flattened $s \times s$ image patches. Set $\boldsymbol{\mu}^j, \sigma$ as in general case.
F^i:	Ftrs: $F^i_{jk} = \exp(\left\|\mathbf{p}^{ik} - \boldsymbol{\mu}^j\right\|_2^2 / 2\sigma^2)$ where $\mathbf{p}^{ik}$ is the k^{th} patch in image i.
U^i, Σ^i:	For each $\mathbf{x}_i$, compute the difference to neighbor matrix Δ^i and apply the SVD to get: $\Delta^i = U^i \Sigma^i {V^i}^\top$

MINIMIZATION

Initialize Θ to a random $d \times f$ matrix.

while $\text{error}_{img}(\theta)$ still decreases **do** $\quad \forall i$, solve for the best E^i given Θ:

$$E^i = \left({F^i}^\top \Theta^\top\right)^+ U^i \Sigma^i$$

solve for the best Θ given the E^i's:

$$A^{ij} = {E^i_{\cdot j}}^\top \otimes {F^i}^\top$$

$$A = \left(\sum_{i=1}^{n} \sum_{j=1}^{D} {A^{ij}}^\top A^{ij}\right)$$

$$\text{vec}\left(\Theta^\top\right) = A^+ \sum_{i=1}^{n} \sum_{j=1}^{D} \Sigma^i_{jj} {A^{ij}}^\top U^i_{\cdot j}$$

4

Behavior Recognition via Sparse Spatio-Temporal Features

In this chapter we develop a general framework for detecting and characterizing behavior from video sequences, making few underlying assumptions about the domain and subjects under observation. Consider some of the well-known difficulties faced in behavior recognition. Subjects under observation can vary in posture, appearance and size. Occlusions and complex backgrounds can impede observation, and variations in the environment, such as in illumination, can further make observations difficult. Moreover, there are variations in the behaviors themselves.

Many of the problems described above have counterparts in object recognition. The inspiration for our approach comes from approaches to object recognition that rely on sparsely detected features in a particular arrangement to characterize an object, e.g. [21, 2, 44]. Such approaches tend to be robust to pose, image clutter, occlusion, object variation, and the imprecise nature of the feature detectors. In short they can provide a robust descriptor for objects without relying on too many assumptions.

We propose to characterize behavior through the use of spatio-temporal feature points (see figure 4.1). A spatio-temporal feature is a short, local video sequence such as an eye opening or a knee bending, or for a mouse, a paw rapidly moving back and forth. A behavior is then fully described in terms of the types and locations of feature points present (for simplicity in this work we ignore the feature

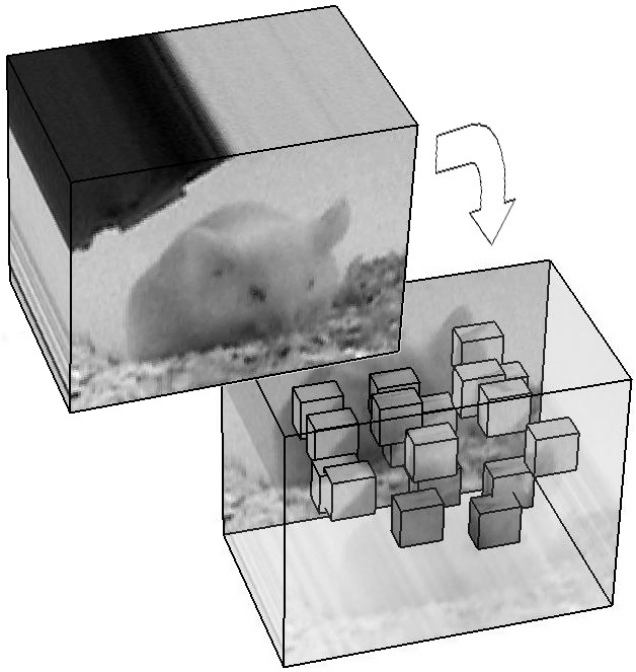


Figure 4.1: Visualization of cuboid based behavior recognition. A spatio-temporal volume of mouse footage shown at top. We apply a spatio-temporal interest point detector to find local regions of interest in space and time (cuboids) which serve as the substrate for behavior recognition.

locations). The motivation is that a particular behavior can be characterized as such regardless of global appearance, posture, nearby motion or occlusion and so forth, for example, see figure 4.2. The complexity of discerning whether two behaviors are similar is shifted to the detection and description of a rich set of features.

Although the method is inspired by approaches to object recognition that rely on spatial features, video and images have distinct properties. The third dimension is temporal, not spatial, and must be treated accordingly. Detection of objects in 3D spatial volumes is a distinct problem, see for example [25].

In this chapter we show that direct 3D counterparts to commonly used 2D interest point detectors are inadequate for detection of spatio-temporal feature points and propose an alternative. We also develop and test a number of descriptors to characterize the cuboids of spatio-temporally windowed data surrounding a feature point. Cuboids extracted from a number of sample behaviors from a given domain are clustered to form a dictionary of cuboid prototypes. The only information kept from all subsequent video data is the location and type of the cuboid prototypes present. We argue that such a representation is sufficient for recognition and robust with respect to variations in the data. We show applications of this framework, utilizing a simple behavior descriptor, to three datasets containing human and mouse behaviors, and show superior results over a number of existing algorithms.

The structure of the chapter is as follows. In Section 4.1 we discuss related work. We describe our algorithm in Section 4.2. In Section 4.3 we present a detailed comparison of the performance of our algorithm versus existing methods on various datasets. We conclude in Section 4.4.

4.1 Related Work

Tracking and behavior recognition are closely related problems, and in fact many traditional approaches to behavior recognition are based on tracking models of varying sophistication, from paradigms that use explicit shape models in either 2D or 3D to those that rely on tracked features; for a broad overview see [26]. The basic idea is that given a tracked feature or object, its time series provides a descriptor

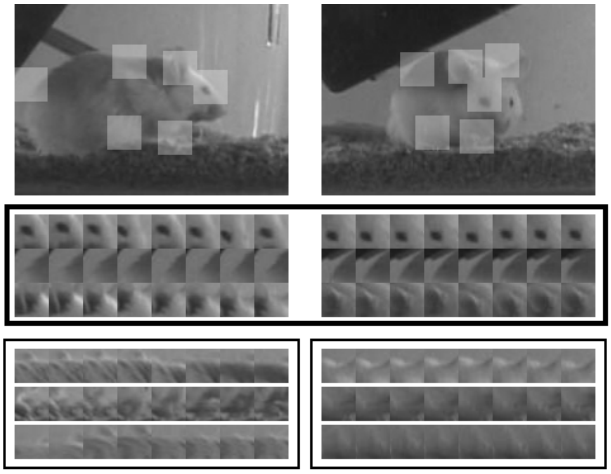


Figure 4.2: Example of six cuboids extracted from two different sequences of grooming. A single frame is shown from each original sequence. Below, each cuboid is shown over time. Note that although the posture of the mouse is quite different in the two cases, three of the six cuboids (shown in the top three rows) for each mouse are quite similar. The other three have no obvious correspondences, although it's very hard to perceive what these are without motion.

that can be used in a general recognition framework.

In the domain of human behavior recognition for example, an entire class of approaches for recognition is based on first recovering the location and pose of body parts, see for example [80, 11]. However, it is unclear how to extend paradigms that rely on articulated models in either 2D or 3D to domains where behavior is not based on changes in configurations of rigid parts, as is the case for recognition of rodent behavior. Perhaps more fundamental, however, is that even in domains where explicit shape models are applicable, it is often very difficult to fit the models to the data accurately.

Another class of approaches performs recognition by first tracking a number of spatial features. Song *et al.* [67] use spatial arrangements of tracked points to distinguish between walking and biking, using the intuition that people can identify such behaviors from Johansson displays. In [56], the authors use view invariant aspects of the trajectory of a tracked hand to differentiate between actions such as opening a cabinet or picking up an object. Recognition can also proceed from

tracked contours, such as in [35].

In response to the practical difficulties of feature and contour tracking, [51] and [69] introduced the framework of 'tracking as repeated recognition,' in which the recovery of pose and body configuration emerges as a byproduct of frame-by-frame recognition using a hand labeled dataset of canonical poses. These approaches are based on the comparison of Canny edges. While the assumptions of edge detection are less restrictive than those of feature or contour tracking, it is still unreliable in domains with cluttered or textured backgrounds or in which the object of interest has poor contrast.

The work of Efros et al. [20] focuses on the case of low resolution video of human behaviors, targeting what they refer to as 'the 30 pixel man.' In this setting they propose a spatio-temporal descriptor based on optical flow measurements, and apply it to recognize actions in ballet, tennis and football datasets. Our proposed method bears some similarity to this approach, but is categorically different in that it uses local features rather than a global measurement. Earlier approaches in this vein are those of [81] and [14]. In [81] for example, Zelnik-Manor and Irani use descriptors based on global histograms of image gradients at multiple temporal scales. The approach shows promise for coarse video indexing of highly visually distinct actions. We examine the approaches of [81] and [20] in more detail in section 4.3.3.

Most closely related to our approach is that of [63], who also use sparsely detected spatio-temporal features for recognition, building on the work on spatio-temporal feature detectors by [43]. They show promising results in human behavior recognition, demonstrating the potential of a method based on spatio-temporal features in a domain where explicit shape models have traditionally been used. The spatio-temporal detector, feature descriptor and behavior descriptor employed in their approach differ from ours. Their method assumes fixed length behaviors, and the similarity between a pair of behaviors is found using a greedy match of the features where multiple features can map to the same corresponding feature. Our approach allows varying length behaviors and is applicable to a much broader range of data.

4.2 Proposed Algorithm

In the following sections we describe our algorithm in detail. In Section 4.2.1 we discuss detection of spatial interest points and extensions to the spatio-temporal domain. We describe cuboids in more detail in Section 4.2.2, and in Section 4.2.3 we describe the use and importance of cuboid prototypes. We describe the very simple behavior descriptor used in all of our experiments in Section 4.2.4.

4.2.1 Feature Detection

A variety of methods exist to detect interest points in the spatial domain, for an extensive review and comparison of methods see [60]. Typically, a response function is calculated at every location in the image and feature points correspond to local maxima.

One of the most popular approaches to interest point detection in the spatial domain is based on the detection of corners, such as [31, 22]. Corners are defined as regions where the local intensity gradient vectors point in orthogonal directions. The gradient vectors are obtained by taking the first order derivatives of a smoothed image $L(x, y, \sigma) = I(x, y) * g(x, y, \sigma)$, where g is the Gaussian smoothing kernel. σ controls the spatial scale at which corners are detected. The response strength at each point is then based on the rank of the covariance matrix of the gradient calculated in a local window. Different measures of the rank lead to slightly different algorithms.

Another common approach is to use the Laplacian of Gaussian (LoG) for the response function. For example, Lowe [45] proposed using an approximation of the LoG based on the difference of the image smoothed at different scales. Specifically, his response function is $D = (g(\cdot; k\sigma) - g(\cdot; \sigma)) * I = L(\cdot; k\sigma) - L(\cdot; \sigma)$ where k is a parameter that controls the accuracy of the approximation; D tends to the scale normalized LoG as k goes to 1. Under varying conditions either the LoG or Harris detector may have better performance; [49] proposes a technique that incorporates both approaches.

Kadir and Brady [37] approach feature detection with the specific goal of detecting features for object recognition. They define a local measure of patch

complexity and look for points the maximize this measure spatially and across scales. Motivation for this type of approach is that salient points are precisely those which maximize discriminability between the objects. This feature detector was used by [21] in their object recognition framework.

Extensions to the Spatio-Temporal Case

The general idea of interest point detection in the spatio-temporal case is similar to the spatial case. Instead of an image $I(x, y)$, interest point detection must operate on a stack of images denoted by $I(x, y, t)$. Localization must proceed not only along the spatial dimensions x and y but also the temporal dimension t. Likewise, detected features also have temporal extent.

The only spatio-temporal interest point operator that we know of is an extension of the Harris corner detect to the 3D case, which has been studied quite extensively by Laptev and Lindeberg (for a recent work see [43]). The basic idea is simple and elegant. Gradients can be found not only along x and y, but also along t, and spatio-temporal corners are defined as regions where the local gradient vectors point in orthogonal directions spanning x, y and t. Intuitively, a spatio-temporal corner is an image region containing a spatial corner whose velocity vector is reversing direction. The second moment matrix is now a 3×3 matrix, and the response function is again based on the rank of this matrix.

The generalized Harris detector described above has many interesting mathematical properties, and in practice it is quite effective at detecting spatio-temporal corners. As mentioned [63] used spatio-temporal features detected by the generalized Harris detector to build a system that distinguishes between certain human behaviors. The behaviors their method can discriminate amongst, including walking, jogging, running, boxing, clapping and waving, are in fact well characterized by the reversal in the direction of motion of arms and legs. Hence these behaviors give rise to spatio-temporal corners, so the technique is well suited for dealing with their dataset.

In certain problem domains, e.g., rodent behavior recognition or facial expressions, we have observed that true spatio-temporal corners are quite rare, even when seemingly interesting motion is occurring. Sparseness is desirable to an extent,

but features that are too rare can prove troubling in a recognition framework, as observed by Lowe [45].

In addition to the rarity of spatio-temporal corners, a more general question that remains unanswered is whether spatio-temporal corners are in fact the features one needs for general behavior recognition. Analogous to useful features for object recognition, we are interested in precisely those features that maximize discrimination between behaviors. Consider two examples, the jaw of a horse chewing on hay and the spinning wheel of a bicycle. Neither example gives rise to a spatio-temporal corner as the motions are subtle and gradually changing, yet both seem like particularly relevant features for behavior recognition.

We propose an alternative spatio-temporal feature detector for our behavior recognition framework. We have explicitly designed the detector to err on the side of detecting too many features rather than too few, noting that object recognition schemes based on spatial interest points deal well with irrelevant and possibly misleading features generated by scene clutter and imperfect detectors [45]. The resulting representation is still orders of magnitude sparser than a direct pixel representation.

Like much of the work on interest point detectors, our response function is calculated by application of separable linear filters. We assume a stationary camera or a process that can account for camera motion. The response function has the form $R = (I * g * h_{ev})^2 + (I * g * h_{od})^2$ where $g(x, y; \sigma)$ is the 2D Gaussian smoothing kernel, applied only along the spatial dimensions, and h_{ev} and h_{od} are a quadrature pair [29] of 1D Gabor filters applied temporally. These are defined as $h_{ev}(t; \tau, \omega) = -\cos(2\pi t\omega)e^{-t^2/\tau^2}$ and $h_{od}(t; \tau, \omega) = -\sin(2\pi t\omega)e^{-t^2/\tau^2}$. In all cases we use $\omega = 4/\tau$, effectively giving the response function R two parameters σ and τ, corresponding roughly to the spatial and temporal scale of the detector. The spatio-temporal filters $g * h_{od}$ and $g * h_{ev}$ take on the form shown in figure 4.3.

The detector is tuned to fire whenever variations in local image intensities contain periodic frequency components. In general there is no reason to believe that only periodic motions are interesting. Periodic motions, such as a bird flapping its wings, will indeed evoke the strongest responses, however, the detector responds strongly to a range of other motions, including spatio-temporal corners. In general,

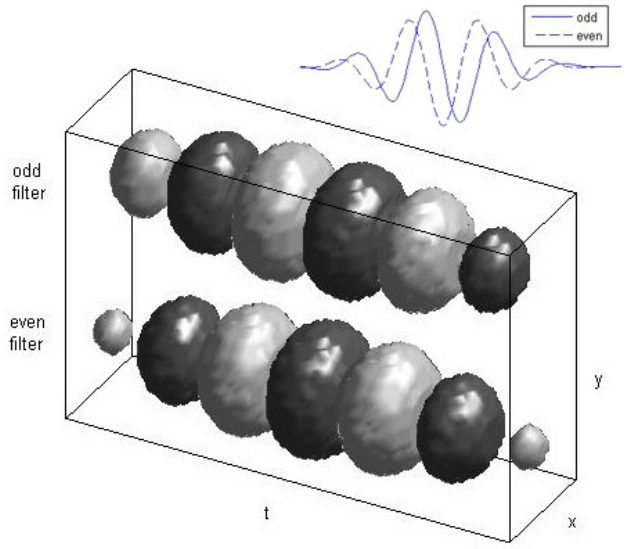


Figure 4.3: Filters used to detect interest points, tuned to fire maximally when intensity in a local spatial region oscillates temporally. Dark lobes correspond to negative areas. Surfaces shown are drawn at 10% of the peak filter response. The 1D profile of the temporal filters is shown in the upper right, the heights of the peaks corresponds to the diameters of the lobes.

any region with spatially distinguishing characteristics undergoing a complex motion can induce a strong response. Areas undergoing pure translational motion will in general not induce a response, as a moving, smoothed edge will cause only a gradual change in intensity at a given spatial location. Areas without spatially distinguishing features cannot induce a response.

4.2.2 Cuboids

At each interest point (local maxima of the response function defined above), a cuboid is extracted which contains the spatio-temporally windowed pixel values. The size of the cuboid is set to contain most of the volume of data that contributed to the response function at that interest point; specifically, cuboids have a side length of approximately six times the scale at which they were detected per spatial and temporal dimension.

To compare two cuboids, a notion of similarity needs to be defined. Given the large number of cuboids we deal with in some of the datasets (on the order of

10^5), we opted to use a descriptor that could be computed once for each cuboid and compare using Euclidean distance.

The simplest cuboid descriptor is a vector of flattened cuboid values. More generally, a transformation can be applied to the cuboid, such as normalization of the pixel values, and given the transformed cuboid, various methods can be employed to create a feature vector, such as histogramming. The goal of both phases is to create a descriptor with invariance to small translations, slight variation in appearance or motion, changes in lighting, and so on, while retaining the descriptor's discriminative power. Instead of trying to predict the right balance between invariance and discriminative power, we design a number of descriptors and test each in our recognition framework.

The transformations we apply to each cuboid include: (1) normalized brightness values (pixels are set to have zero mean and unit variance), (2) the brightness gradient, and (3) windowed optical flow. The brightness gradient is calculated at each spatio-temporal location (x, y, t), giving rise to three channels (G_x, G_y, G_t) each the same size as the cuboid. To extract motion information we calculate Lucas-Kanade optical flow [46] between each pair of consecutive frames, creating two channels (V_x, V_y). Each channel is the same size as the cuboid, minus one frame.

We use one of three methods to create a feature vector given the transformed cuboid (or multiple resulting cuboids when using the gradient or optical flow). The simplest method involves flattening the cuboid into a vector, although the resulting vector is potentially sensitive to small cuboid perturbations. The second method involves histogramming the values in the cuboid. Such a representation is robust to perturbations but also discards all positional information (spatial and temporal). Local histograms, used as part of Lowe's 2D SIFT descriptor [45], provide a compromise solution. The cuboid is divided into a number of regions and a local histogram is created for each region. The goal is to introduce robustness to small perturbations while retaining some positional information. For all the methods, to reduce the dimensionality of the final descriptors we use PCA [32].

Many of the above choices were motivated by research in descriptors for 2D features (image patches). For a detailed review of 2D descriptors see [50]. Other

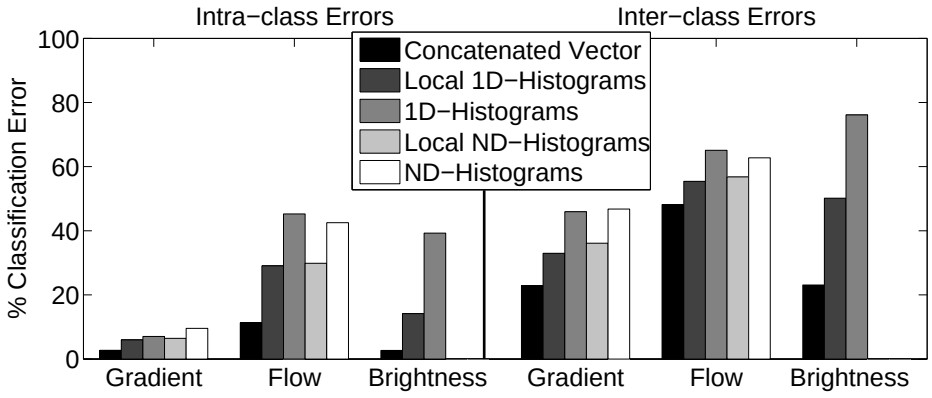


Figure 4.4: Shown is the intra and inter class performance of our recognition method on the face dataset using different cuboid descriptors. The full algorithm, dataset and methodology are discussed later, the sole purpose of this figure is to give a sense of the relative performance of the various cuboid descriptors. Recall that the descriptors we use involve first transforming the cuboid into: (1) normalized brightness, (2) gradient, or (3) windowed optical flow, followed by a conversion into a vector by (1) flattening, (2) global histogramming, or (3) local histogramming, for a total of nine methods, along with multi-dimensional histograms when they apply. Using the gradient in any form gave very reliable results, as did using the flattened vector of normalized brightness values.

spatio-temporal descriptors are possible. For example, Schuldt *et al*. [63] used differential descriptors [40] for their spatio-temporal interest points, however, among the descriptors examined for 2D features, differential descriptors are not particularly robust.

We tested the performance of our overall algorithm changing only the cuboid descriptor on a dataset described later in this chapter. Results are shown in figure 4.4. Histograms, both local and global did not provide improved performance; apparently the added benefit of increased robustness was offset by the loss of positional information. In all experiments reported later in the chapter we used the flattened gradient as the descriptor, which is essentially a generalization of the PCA-SIFT descriptor [38].

4.2.3 Cuboid Prototypes

Our approach is based on the idea that although two instances of the same behavior may vary significantly in terms of their overall appearance and motion, many of the interest points they give rise to are similar. Under this assumption, even though the number of possible cuboids is virtually unlimited, the number of different *types* of cuboids is relatively small. In terms of recognition the exact form of a cuboid becomes unimportant, only its type matters.

We create a library of cuboid prototypes by clustering a large number of cuboids extracted from the training data. We cluster using the k-means algorithm. The library of cuboid prototypes is generated separately for each dataset since the cuboids types are very different in each (mouse cuboids are quite distinct from face cuboids). Clusters of cuboids tend to be perceptually meaningful.

Using cluster prototypes is a very simple yet powerful method for reducing variability of the data while maintaining its richness. After the training phase, each cuboid detected is either assumed to be one of the known types or rejected as an outlier.

Intuitively the prototypes serve a similar function as parts do in object recognition. The definition of parts varies widely in the literature on object recognition, the analogy here is most applicable to the work of [21] and especially [2], who refer to the local neighborhoods of spatially detected interest points as parts. In the

case of static face detection, these might include the eyes or hairline features.

4.2.4 Behavior Descriptor

After extraction of the cuboids the original clip is discarded. The rationale for this is that once the interest points have been detected, together their local neighborhoods contain all the information necessary to characterize a behavior. Each cuboid is assigned a type by mapping it to the closest prototype vector, at which point the cuboids themselves are discarded and only their type is kept.

We use a histogram of the cuboid types as the behavior descriptor. Distance between the behavior descriptors (histograms) can be calculated by using the Euclidean or χ^2 distance. When more training data is available, we use the behavior descriptor and class labels in a classification framework.

The relative positions of the cuboids are currently not used. Previously mentioned algorithms for object recognition, such as [21] or [2] could be used as models for how to incorporate positional information.

4.3 Experiments

We explore results in three representative domains: facial expressions, mouse behavior and human activity. Representative frames are shown in figure 4.5. To judge the performance of our algorithm, we compare to results obtained using three other general activity recognition algorithms on these datasets. Each domain presents its own challenges and demonstrates various strengths and weaknesses of each algorithm tested.

We describe each dataset in detail in the following section, training and testing methodology in Section 4.3.2, the algorithms used for comparison in Section 4.3.3, and finally detailed results in Section 4.3.4.

4.3.1 Datasets

We compiled the facial expressions and mouse behavior datasets ourselves, they are available for download at `http://vision.ucsd.edu`. The human activity

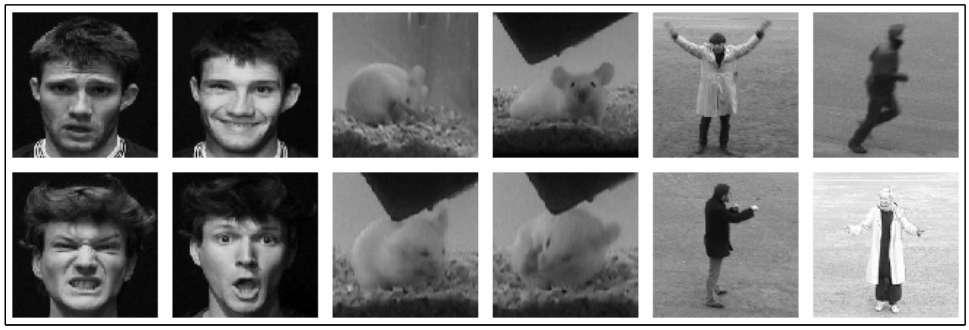


Figure 4.5: Representative frames from clips in each domain: (a) facial expressions, (b) mouse behavior, and (c) human activity.

dataset was collected by [63] and is available online at
`http://www.nada.kth.se/cvap/actions/`.

The face data involves 2 individuals, each expressing 6 different emotions under 2 lighting setups. The expressions are anger, disgust, fear, joy, sadness and surprise. Certain expressions are quite distinct, such as sadness and joy, others are fairly similar, such as fear and surprise. Under each lighting setup, each individual was asked to repeat each of the 6 expressions 8 times. The subject always starts with a neutral expression, expresses an emotion, and returns to neutral, all in about 2 seconds.

The mouse data includes short clips taken from seven fifteen minute videos of the same mouse filmed at different points in the day. Each clip contains the mouse engaged in a single behavior. The set of behaviors includes drinking, eating, exploring, grooming and sleeping. The number of occurrences and characteristics of each behavior vary substantially for each of the seven videos. A total of 406 clips were extracted ranging from 14 occurrences of drinking to 159 occurrences of exploring, each lasting between 1 and 10 seconds. Typical mouse diameter is approximately 120 pixels although the mouse can stretch or compress substantially. All filming was done in the vivarium in which the mice are housed. The videos were collected with help from veterinarians at the UCSD Animal Care Program, who also advised on how to classify and label the data by hand.

In order to be able to do a full comparison of methods, we also created a

greatly simplified, small scale version of the mouse dataset. While the mouse eats, it tends to sit still, and on occasion when it explores it sniffs around but remains stationary. From two different mouse videos we extracted a number of examples of these two behaviors, all of the same (short) duration, and made sure the mouse is spatially centered in each. Data in this form does not benefit our algorithm in any way, however, it is necessary to get results for some of the methods we test against.

The human activity data comes from the dataset collected by [63]. There are 25 individuals engaged in the following activities: walking, jogging, boxing, clapping and waving. We use a subset of the dataset which includes each person repeating each activity 8 times for about 4 seconds each, wearing different clothing (referred to scenarios $s1$ and $s3$), for a total of almost 1,200 clips. The clips have been sub-sampled (people are approximately 80 pixels in height) and contain compression artifacts (this is the version of the dataset available online).

4.3.2 Methodology

We divide each dataset into groups. The groups we chose for the datasets discussed above are as follows: face clips are divided into 4 groups, one group per person per lighting setup; mouse clips are divided into 7 groups, corresponding to each of the source videos; human activity clips are divided into 25 groups, one per person. We analyze the performance of various algorithms trained on a subset of the groups and tested on a different subset. Often, because of the limited amount of data, we use leave one out cross validation to get an estimate of performance.

All algorithms have parameters that need tuning. In all cases that we report results we report the best performance achieved by a given algorithm – parameter sweeps were done for all the algorithms. As can be seen in figure 4.6 our method is not very sensitive to the exact parameter settings, in fact, aside from the scale of the cuboids we used the same parameter settings on all three datasets. Some of the algorithms also have a random component (for example a clustering phase), in this case any experiment reported is averaged over 20 runs.

When applicable, we focus on reporting relative performance of the algorithms so as to avoid questions of the absolute difficulty of a given dataset.

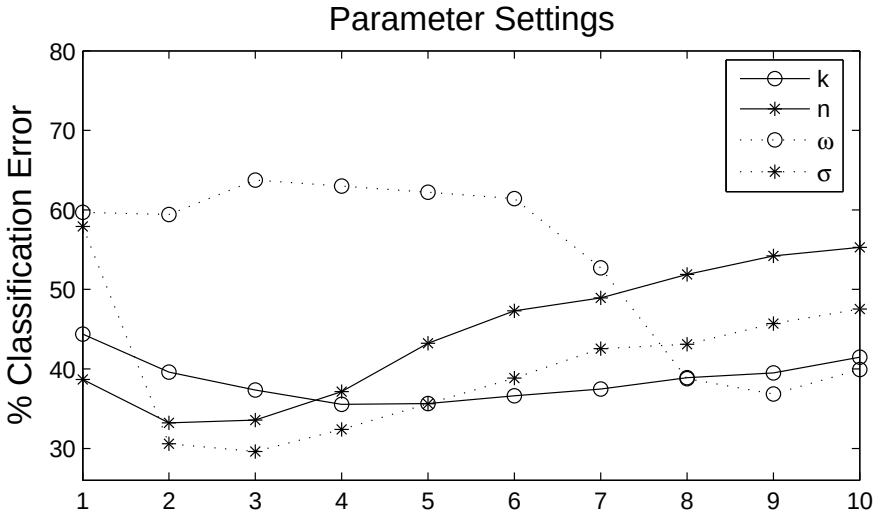


Figure 4.6: We tested how sensitive the performance of our method is to various parameter settings on the face dataset. In each of the above curves we plot classification error for 10 different settings of a given parameter with all other parameters kept constant at default, 'reasonable' values. Random guessing would result in 80% error; all data points shown fall well below this. The x-axis differs for each parameter; the thing to note is that the overall shape of each curve is smooth and tends to be bowl shaped. The four parameters shown are: k, $50 < k < 500$, the number of clusters prototypes (*i.e.* cuboid types), n, $10 \leq n \leq 200$ the number of cuboids detected per face clips, ω, $0 < \omega < 1$ the fraction overlap allowed between cuboids, and σ, $.2 < \sigma < 9$, the spatial scale of the detector (which also determines the size of the cuboid). Optimal settings were approximately: $k = 250$, $n = 30$, $\omega = .9$ and $\sigma = 2$.

4.3.3 Algorithms for Comparison

We compare our approach to three other methods. Each of these is a general purpose behavior recognition algorithm that is capable of dealing with low resolution and noisy data. We implement the algorithms of Efros et al. [20] and Zelnik-Manor and Irani [81], we refer to these as EFROS and ZMI, respectively. We also use a variation of our framework based on the Harris 3D corner detector [63], described previously. The only difference between this and our framework is that we swap our feature detector for the Harris corner detector. We refer to our framework as CUBOIDS and to the variation using the Harris detector as CUBOIDS+HARRIS[1]. Unless otherwise specified we use 1-nearest neighbor classifier with the χ^2 distance on top of the cuboid representation. We describe EFROS and ZMI in more detail below.

EFROS is used to calculate the similarity of the activity of two subjects using a version of normalized cross correlation on optical flow measurements. Subjects must be tracked and stabilized. If the background is non uniform this can also require figure-ground segmentation. However, when these requirements are satisfied the method has been shown to work well for human activity recognition and has been tested on ballet, tennis and football datasets[2]. EFROS tends to be particularly robust to changes in appearance and has shown impressive results even on very low resolution video.

ZMI works by histogramming normalized gradient measurements from a spatio-temporal volume at various temporal scales, resulting in a coarse descriptor of activity. No assumptions are made about the data nor is tracking or stabilization required. The method's strength lies in distinguishing motions that are grossly different; promising results have been shown on human activities such as running, waving, rolling or hopping. In some sense ZMI and EFROS are complementary algorithms and we could expect one to perform well when the other does not.

[1]This algorithm is very different from the work of [63], the only similarity is that both use features detected by the Harris corner detector.

[2]Unfortunately, these datasets are no longer available.

4.3.4 Results

In the following sections we show results on the datasets described above: facial expressions, human activity and mouse behavior. In all experiments on all datasets, CUBOIDS had the highest recognition rate, often by a wide margin. Typically the error is reduced by at least a third from the second best method.

Facial Expression

In each experiment, training is done on a single subject under one of the two lighting setups and tested on: (1) the same subject under the same illumination[3], (2) the same subject under different illumination, (3) a different subject under the same illumination, and (4) a different subject under different illumination. Results are shown in figure 4.7. In all cases CUBOIDS had the highest recognition rates. A majority of the error is caused by anger being confused with other expressions. Subjectively, the two subjects' expression of anger is quite different, which may account for this phenomenon.

Mouse Behavior

The mouse data presents a highly challenging behavior recognition problem. Differences between behaviors can be subtle, optical flow calculations tend to be inaccurate, the mouse blends in with the bedding of the cage, and there are no easily trackable features on the mice themselves (the eyes of the mouse are frequently occluded or closed). The pose of the mouse w.r.t. the camera also varies significantly.

Results on the full dataset are presented in figure 4.8, on the left. The overall recognition rate is around 72%. As mentioned, we also used a simplified, small scale version of the mouse dataset in order to do a full comparison of methods[4]. In both experiments CUBOIDS had the lowest errors, see figure 4.8, on the right.

[3] In this case we use leave one out cross validation.

[4] EFROS requires a stabilized figure, and with a non-uniform background stabilization requires figure-ground segmentation, a non-trivial task.

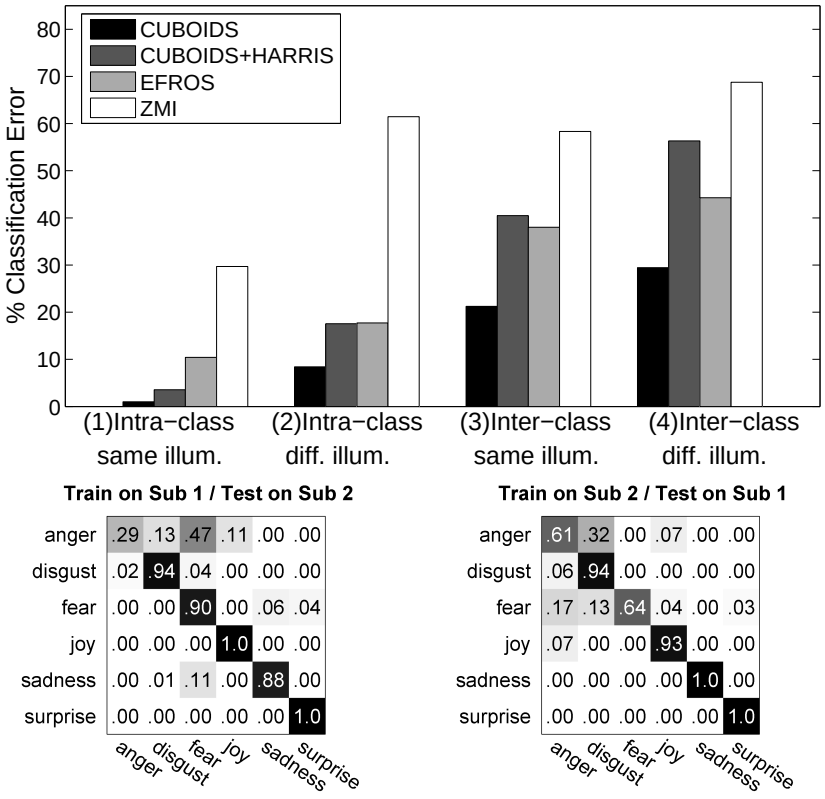


Figure 4.7: FACE DATASET *Top row:* We investigated how identity and lighting affect each algorithm's performance. In all cases CUBOIDS gave the best results. EFROS and CUBOIDS+HARRIS had approximately equal error rates, except that EFROS tended to perform better under changes in illumination. ZMI was not well suited to discriminating between facial expressions, performing only slightly better than chance. Random guessing would result in 83% error. All algorithms were run with optimal parameters. *Bottom row:* Inter-class confusion matrices obtained using our method under the first illumination setup on the face data. "Sub 1" and "Sub 2" refer to the first and second subjects, respectively. A majority of the error is caused by anger being confused with other expressions. Subjectively, the two subjects' expression of anger is quite different.

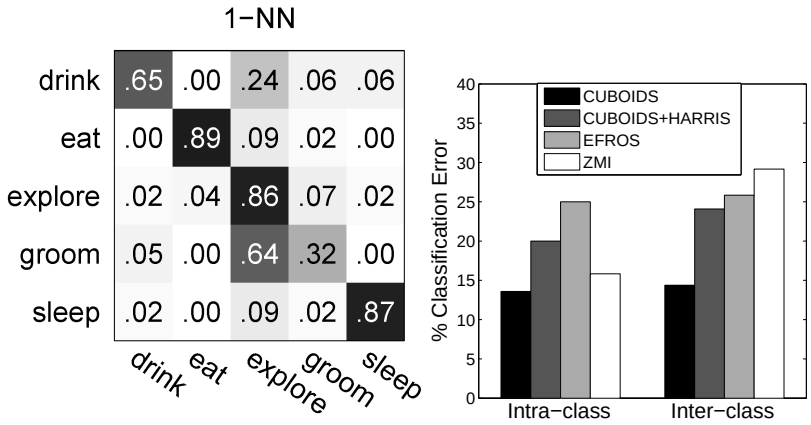


Figure 4.8: MOUSE DATASET *Left:* Confusion matrix generated by CUBOIDS on the full mouse dataset. As mentioned, this dataset presents a challenging recognition problem. Except for a few difficult categories, recognition rates using our method were fairly high. *Right:* Due to the form of the data, a full comparison of algorithms was not possible. Instead, we created a simple small scale experiment and ran all four algorithms on it. CUBOIDS had the lowest error rates, ZMI was a near second on intra-class error. Note that all algorithms did far better than chance, which would result in 80% error.

Human Activity

For the human activity dataset we used leave one out cross validation to get the overall classification error. Due to the large size of this dataset, we did not attempt a comparison with other methods[5]. Rather, we provide results only to show that our algorithm works well on a diverse range of data. Confusion matrices for the six categories of behavior are shown in figure 4.9; the overall recognition rate was over 80%.

4.4 Discussion

In this chapter we have shown the viability of doing behavior recognition by characterizing behavior in terms of spatio-temporal features. A new spatio-temporal interest point detector was presented, and a number of cuboid descriptors were analyzed. We showed how the use of cuboid prototypes gave rise to an efficient

[5]Although the confusion matrices in figure 4.9 are better than those reported in [63], the results are not directly comparable because the methodologies are different.

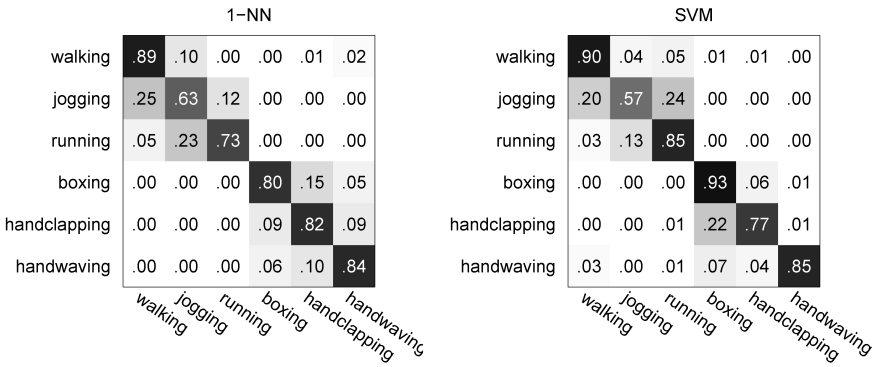


Figure 4.9: HUMAN ACTIVITY DATASET Shown are confusion matrices generated by CUBOIDS. Two classifiers were used: 1-nearest neighbor and Support Vector Machines with radial basis functions [32]. Using SVMs resulted in a slight reduction of the error. Note that most of the confusion occurs between jogging and walking or running, and between boxing and clapping, most other activities are easily distinguished. The overall error of our approach was about 20% compared to random guessing which would result in 83% error.

and robust behavior descriptor. Throughout we have tried to establish the link between the domains of behavior recognition and object recognition, creating the potential to bring in a range of established techniques from the spatial domain to that of behavior recognition.

We have demonstrated our framework across a number of domains and obtained state of the art results; in all experiments outperforming well established algorithms. Our approach avoids the challenges of tracking, background-foreground modeling, model fitting, *etc*. The proposed method is computationally efficient: our Matlab implementation is real time[6]. Furthermore the approach is straightforward and easy to implement.

Future extensions include using the spatio-temporal layout of the features, extending such approaches as [3] or [2] to the spatio-temporal domain. Using features detected at multiple scales should also improve performance. Another possible direction of future work is to incorporate a dynamic model on top of our representation.

Portions of this chapter are based on "Behavior Recognition via Sparse

[6]In our implementation we require the entire video to be stored on disk, but this is not a fundamental requirement of the algorithm.

Spatio-Temporal Features" by P. Dollár, V. Rabaud, G. Cottrell and S. Belongie, [16]. I was responsible for the development of the algorithm, experimental design and literature survey, and also wrote most of the paper.

Bibliography

[1] E. Di Bernardo, L. Goncalves and P. Perona.US Patent 6,552,729: Automatic generation of animation of synthetic characters., 2003.

[2] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, Nov 2004.

[3] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1300–1305, 1997.

[4] Y. Bengio and M. Monperrus. Non-local manifold tangent learning. In *Advances in Neural Information Processing Systems*. 2005.

[5] Y. Bengio, J. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet. Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems*, 2004.

[6] D. Beymer and T. Poggio. Face recognition from one example view. In *Proceedings of the IEEE International Conference on Computer Vision*, page 500, Washington, DC, USA, 1995. IEEE Computer Society.

[7] V. Blanz and T. Vetter. Face recognition based on fitting a 3D morphable model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1063–1074, 2003.

[8] M. Brand. Charting a manifold. In *Advances in Neural Information Processing Systems*, 2003.

[9] K. Branson and S. Belongie. Tracking multiple mouse contours (without too many samples). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[10] C. Bregler, M. Covell, and M. Slaney. Video rewrite: driving visual speech with audio. In *Special Interest Group for Computer Graphics*, pages 353–360, 1997.

[11] C. Bregler, J. Malik, and K. Pullen. Twist based acquisition and tracking of animal and human kinematics. *International Journal of Computer Vision*, 56(3):179–194, Feb 2004.

[12] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Nov. 1986.

[13] E. Chuang, H. Deshpande, and C. Bregler. Facial expression space learning. In *Pacific Graphics*, 2002.

[14] J. Davis and A. Bobick. The representation and recognition of action using temporal templates. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 928–934, 1997.

[15] P. Dollár, V. Rabaud, and S. Belongie. Learning to traverse image manifolds. In *Advances in Neural Information Processing Systems*, Dec. 2006.

[16] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, October 2005.

[17] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2006.

[18] B. Dubuc and S. Zucker. Complexity, confusion, and perceptual grouping. *International Journal of Computer Vision*, 2001.

[19] G. J. Edwards, T. F. Cootes, and C. J. Taylor. Face recognition using active appearance models. *Proceedings of the European Conference on Computer Vision*, 1998.

[20] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 726–733, Nice, France, 2003.

[21] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[22] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points. In *Intercommission Conf. on Fast Processing of Photogrammetric Data*, pages 281–305, Switzerland, 1987.

[23] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the International Conference on Machine Learning*, 1996.

[24] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Stanford, 1998.

[25] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *Proceedings of the European Conference on Computer Vision*, 2004.

[26] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, January 1999.

[27] B. Geisler. Perceptual grouping and the bayesian statistics of natural scenes. In *Bayes*, 2001.

[28] D. Geman and B. Jedynak. An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996.

[29] G. Granlund and H. Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995.

[30] C. Guo, S. Zhu, and Y. Wu. A mathematical theory of primal sketch and sketchability. In *Proceedings of the IEEE International Conference on Computer Vision*, 2003.

[31] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. Alvey Conf.*, pages 189–192, 1988.

[32] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer Verlag, Basel, 2001.

[33] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[34] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. In *Special Interest Group for Computer Graphics*, 2001.

[35] M. Isard and A. Blake. A mixed-state Condensation tracker with automatic model switching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 107–112, 1998.

[36] N. Jojic, B. Frey, and A. Kannan. Epitomic analysis of appearance and shape. In *Proceedings of the IEEE International Conference on Computer Vision*, 2003.

[37] T. Kadir and M. Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83–105, Nov 2001.

[38] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 506–513, 2004.

[39] D. Keysers, W. Macherey, J. Dahmen, and H. Ney. Learning of variability for invariant statistical pattern recognition. *European Conference on Machine Learning*, 2001.

[40] J. Koenderink and A. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55(6):367–75, 1987.

[41] K. Koffka. *Principles of Gestalt Psychology*. Springer Series in Statistics. Harcourt, Brace and company, New York, 1935.

[42] S. Konishi, A. Yuille, J. Coughlan, and S. Zhu. Statistical edge detection: Learning and evaluating edge cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jan. 2003.

[43] I. Laptev and T. Lindeberg. Space-time interest points. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 432–439, 2003.

[44] B. Leibe and B. Schiele. Scale invariant object categorization using a scale-adaptive mean-shift search. In *Proceedings of the Symposium of the German Association for Pattern Recognition (DAGM)*, Aug. 2004.

[45] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.

[46] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[47] D. Marr. *Vision*. 1982.

[48] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[49] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proceedings of the IEEE International Conference on Computer Vision*, pages I: 525–531, 2001.

[50] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages II: 257–263, 2003.

[51] G. Mori and J. Malik. Estimating human body configurations using shape context matching. In *Proceedings of the European Conference on Computer Vision*, page III: 666 ff., 2002.

[52] T. Moriyama, T. Kanade, J. Xiao, and J. F. Cohn. Meticulously detailed eye region model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.

[53] H. Murase and S. Nayar. Visual learning and recognition of 3D objects from appearance. *International Journal of Computer Vision*, 1995.

[54] J. Niebles and L. Fei-Fei. A hierarchical model of shape and appearance for human action classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, July 2007.

[55] V. Ramesh. *Performance Characterization of Image Understanding Algorithms*. PhD thesis, University of Washington, 1996.

[56] C. Rao and M. Shah. View-invariance in action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages II:316–322, 2001.

[57] R. Rao and D. Ruderman. Learning Lie groups for invariant visual perception. In *Advances in Neural Information Processing Systems*, 1999.

[58] X. Ren, C. Fowlkes, and J. Malik. Scale-invariant contour completion using conditional random fields. In *Proceedings of the IEEE International Conference on Computer Vision*, 2005.

[59] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 2003.

[60] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, June 2000.

[61] A. Schödl, R. Szeliski, D. Salesin, and I. Essa. Video textures. In *Special Interest Group for Computer Graphics*, 2000.

[62] B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 1998.

[63] C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: a local SVM approach. In *Proceedings of the International Conference on Pattern Recognition*, pages III: 32–36, 2004.

[64] P. Simard, Y. LeCun, and J. S. Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems*, 1993.

[65] P. Simard, Y. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural Networks: Tricks of the Trade*, 1998.

[66] M. Singh. Modal and amodal completion generate different shapes. *Psychological Science*, 2004.

[67] Y. Song, L. Goncalves, and P. Perona. Unsupervised learning of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):814–827, 2003.

[68] J. Sullivan and S. Carlsson. Recognizing and tracking human action. In *Proceedings of the European Conference on Computer Vision*, 2002.

[69] J. Sullivan and S. Carlsson. Recognizing and tracking human action. In *Proceedings of the European Conference on Computer Vision*, page I: 629, 2002.

[70] C. Taylor and D. Kriegman. Structure and motion from line segments in multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[71] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290, 2000.

[72] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.

[73] Z. Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, 2005.

[74] Z. Tu, X. Chen, A. Yuille, and S. Zhu. Image parsing: Unifying segmentation, detection, and object recognition. *International Journal of Computer Vision*, 2005.

[75] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, May 1991.

[76] P. Viola and M. Jones. Robust real time object detection. In *Statistical and Computational Theories of Vision*, 2001.

[77] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[78] L. Williams and D. Jacobs. Stochastic completion fields: A neural model of illusory contour shape and salience.

[79] S. Wong, T. Kim, and R. Cipolla. Learning motion categories using both semantic and structural information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, July 2007.

[80] Y. Yacoob and M. Black. Parameterized modeling and recognition of activities. *Computer Vision and Image Understanding*, 73(2):232–247, February 1999.

[81] L. Zelnik-Manor and M. Irani. Event-based analysis of video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages II:123–130, 2001.

[82] Z. Zhang and Zha. Local linear smoothing for nonlinear manifold learning. Technical report, 2003.