Contour and Texture for Visual Recognition of Object Categories

JAMIE DANIEL JOSEPH SHOTTON

Queens' College

UNIVERSITY OF CAMBRIDGE

This dissertation is submitted for the degree of Doctor of Philosophy.

March 2007

ABSTRACT

The recognition of categories of objects in images has become a central topic in computer vision. Automatic visual recognition systems are rapidly becoming central to applications such as image search, robotics, vehicle safety systems, and image editing. This work addresses three sub-problems of recognition: image classification, object detection, and semantic segmentation. The task of classification is to determine whether an object of a particular category is present or not. Object detection aims to localize any objects of the category. Semantic segmentation is a more complete image understanding, whereby an image is partitioned into coherent regions that are assigned meaningful class labels. This thesis proposes novel discriminative learning approaches to these problems.

Our primary contributions are threefold. Firstly, we demonstrate that the contours (the outline and interior edges) of an object are, alone, sufficient for accurate visual recognition. Secondly, we propose two powerful new feature types: (i) a learned codebook of contour fragments matched with an improved oriented chamfer distance, and (ii) a set of texture-based features that simultaneously exploit local appearance, approximate shape, and appearance context. The efficacy of these new features types is evaluated on a wide variety of datasets. Thirdly, we show how, in combination, these two largely orthogonal feature types can substantially improve recognition performance above that achieved by either alone.

Declaration

This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text.

Acknowledgements

Firstly, I must express my warmest gratitude to my supervisors, Roberto Cipolla and Andrew Blake, for their superb guidance. I could not have asked for better support, advice, or enthusiasm, and working with them has been a great pleasure.

Both the Machine Intelligence Laboratory and Microsoft Research, at which I was fortunate enough to be an intern, are filled with the most intellectually stimulating, helpful, and affable people that I have ever worked with. It would take a thesis in itself to thank individually everyone that has helped me through the course of my PhD. Nevertheless, I do wish to extend my thanks to a few individuals, at the peril of inadvertent omission. The work in Chapter 3 stems from an exciting collaboration with John Winn, Carsten Rother, and Antonio Criminisi. For their assistance with various experiments, I am very grateful to Piotr Dollár, Giovanni Maria Farinella, Vittorio Ferrari, and Andreas Opelt. My time at Cambridge would have been far less creative and enjoyable without the camaraderie of Gabe, Matt, Ollie, Julia, Björn, George, Carlos, Julien, Tae-Kyun, Martin, Ram, Oggie, Nanthan, Stefano, Shu-Fai, Neill, and Tom.

I am immensely grateful to Microsoft Research Cambridge for their generous financial support, without which my studies would not have been possible. I would also like to thank the Cambridge University Engineering Department and Queens' College for supporting my attendance at conferences.

Last, but by no means least, my heartfelt thanks to my family for their never-faltering encouragement, support, and love. I would especially like to thank my father for his great help proof-reading.

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 Objective

This thesis proposes new techniques for the automatic recognition of categories of objects in images. We address three sub-goals of recognition: *image classification*, *categorical object detection*, and *semantic segmentation*. These tasks, illustrated in Figure 1.1, are defined as follows:

Image classification aims to group together images containing similar objects, such as horses and airplanes. There may also be a number of *background* images that contain none of the objects under consideration.

Categorical object detection addresses determining the number of instances of a particular object category in an image, and localizing those instances in space and scale.

Semantic segmentation aims to segment an image into semantically coherent regions, and simultaneously assign a class label to each region.

The term *category* will be used throughout this document synonymously with *class*, to denote a particular type of object. A class may contain discrete and structured objects, such as cars and faces, or more amorphous entities such as grass and sky.

1.2 Motivation

Vision has evolved as one of our most important senses. Even deprived of the additional cues of balance, sound, known location, etc., that we use to aid visual perception of the real world, understanding a complex photograph is usually an effortless task. Carefully crafted images, such as those in Figure 1.2, do however trip us up occasionally. We sub-consciously

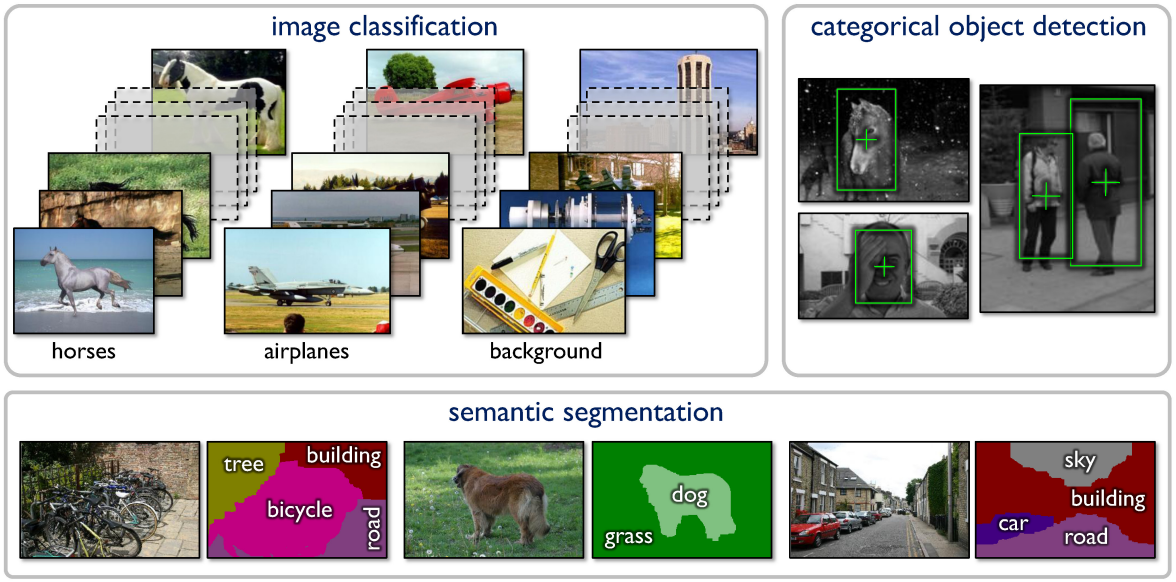

Figure 1.1: **Visual recognition sub-goals.** See text for definitions of terms. The examples of detection and segmentation are actual results from Chapters 2 and 3 respectively.

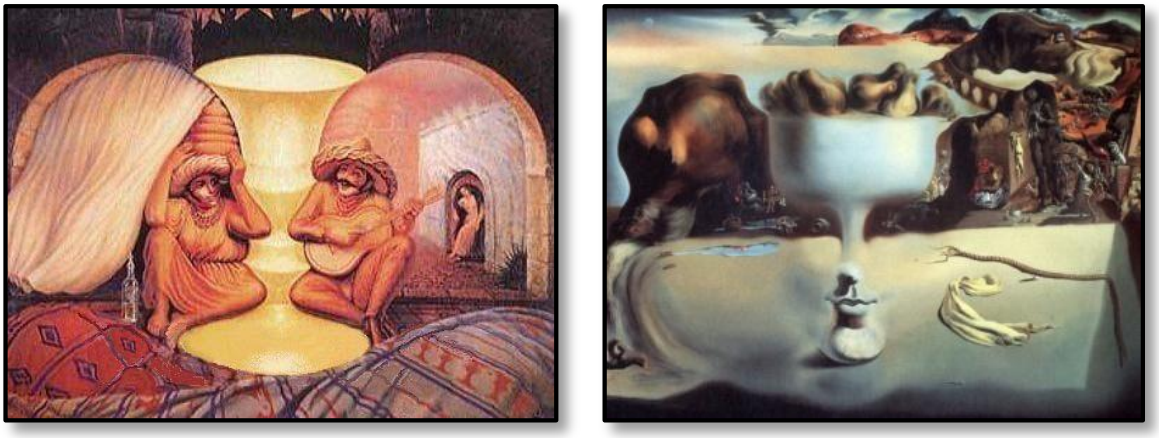

Figure 1.2: **Optical illusions and surrealist art.** Image understanding is usually, but not always, straightforward for the human visual system. While the brain can understand each of these images at a local level, parsing the scene is difficult and ambiguous. [**Left:** "Forever Always" by Octavio Ocampo. **Right:** "Apparition of Face and Fruit Dish on a Beach" by Salvador Dalí].

exploit knowledge of the world around us, learned over many years, to give us context within which to interpret images. The particular sub-goals of recognition that we address in this thesis (classification, detection and segmentation) are clearly achievable by human endeavor, though are impractical across large image databases and the world wide web.

The human visual system is one inspiration for our investigation of computer-based visual recognition, although we do not attempt to directly emulate it in approach, merely in outcome. Motivation beyond that of pure scientific curiosity is provided by several important applications, which in many cases are only becoming feasible with recent advances in the field:

Image search: The world-wide web contains vast quantities of information. Textual information is reliably indexed by search engines such as Google, Live Search, and Yahoo! [Google; Live; Yahoo], allowing almost instantaneous access to billions of documents worldwide. Image search however is still at a nascent stage. Random, accurate semantic access to images and videos on the web would find use in many areas, such as scientific research, illustrating documents and news reports, and for simple web exploration. The current generation of image search engines is based only on metadata and textual cues associated with the image, rather than the image content itself, and this so-called *semantic gap* leads to many incorrect results. Clearly, exploiting the appearance of the image should significantly improve matters.

Medicine: Discovering tumors and other abnormalities in medical scans is an intensive and skilled task. A carefully designed automatic system (e.g. [Cootes & Taylor, 2001]) may be able to both speed up diagnosis and cut down on human error.

Robotics: The field of robotics has advanced dramatically over the last few years. Already, the control systems of humanoid robots allow them to walk and run. However, their usefulness is severely limited without real-time visual understanding of the world that they inhabit. See e.g. [Davison *et al.*, 2007; Se *et al.*, 2005].

Security: Accurate automatic recognition of particular individuals or suspicious behavior could detect nefarious activity in public spaces. Home security systems could also benefit, for example to differentiate between a cat and a burglar in an alarm system.

Transportation: Much current effort is being directed towards improving the safety systems of vehicles, for example, by automatically alerting the driver to pedestrians and

other potential hazards. Automatic license plate and car model recognition has been used to enforce traffic restrictions with the aim of improving road safety.

Image editing: As we describe in more detail at the end of Chapter 3, a semantic understanding of images enables the user interface to be attuned to the semantic class of the region being edited, so that, for example, a gray sky could automatically be made more blue, or the background of a scene could be defocused to concentrate attention on the foreground.

Note that the degree of accuracy required and the consequences of mistakes for these different applications vary considerably. Clearly great care must be taken deploying automatic systems in critical or sensitive applications.

1.3 Sources of Visual Variability

Lacking our human high-level knowledge of visual semantics, computer-based recognition systems face a daunting challenge. We illustrate in Figure 1.3 some of the particular sources of the extreme visual variability that images of objects present due to changes in viewing angle, lighting, scale, and object pose, partial occlusions, and environmental factors. Furthermore, for *categorical* recognition, commonalities must be found to generalize across the variability *within* the class, while determining differences to discriminate *between* classes, as illustrated in Figure 1.4. In this thesis we specifically address the within-class vs. between-class variability, changes in scale, object articulation, and to some extent, lighting and viewing angle. Other work has focused on coping with partial occlusion, e.g. [Winn & Shotton, 2006]. For the tasks of classification and detection we shall assume a particular viewing angle (e.g. side-on), although this assumption is relaxed when investigating semantic segmentation in Chapter 3.

1.4 Approach

We take a modern approach to visual recognition, aiming to *learn* from a set of training images the within-class commonalities, and the between-class differences, that enable us to generalize to recognizing unseen test images. To this end, we employ proven and efficient machine learning techniques, specifically various discriminative classifiers (see Appendix B and Section 3.3).

Figure 1.3: **Sources of visual variability.** Automatic visual recognition systems must deal with visual variability arising from the viewing angle, the pose and articulation of the object, the lighting of the scene, partial occlusions obscuring the object, widely varying scales, and environmental conditions.

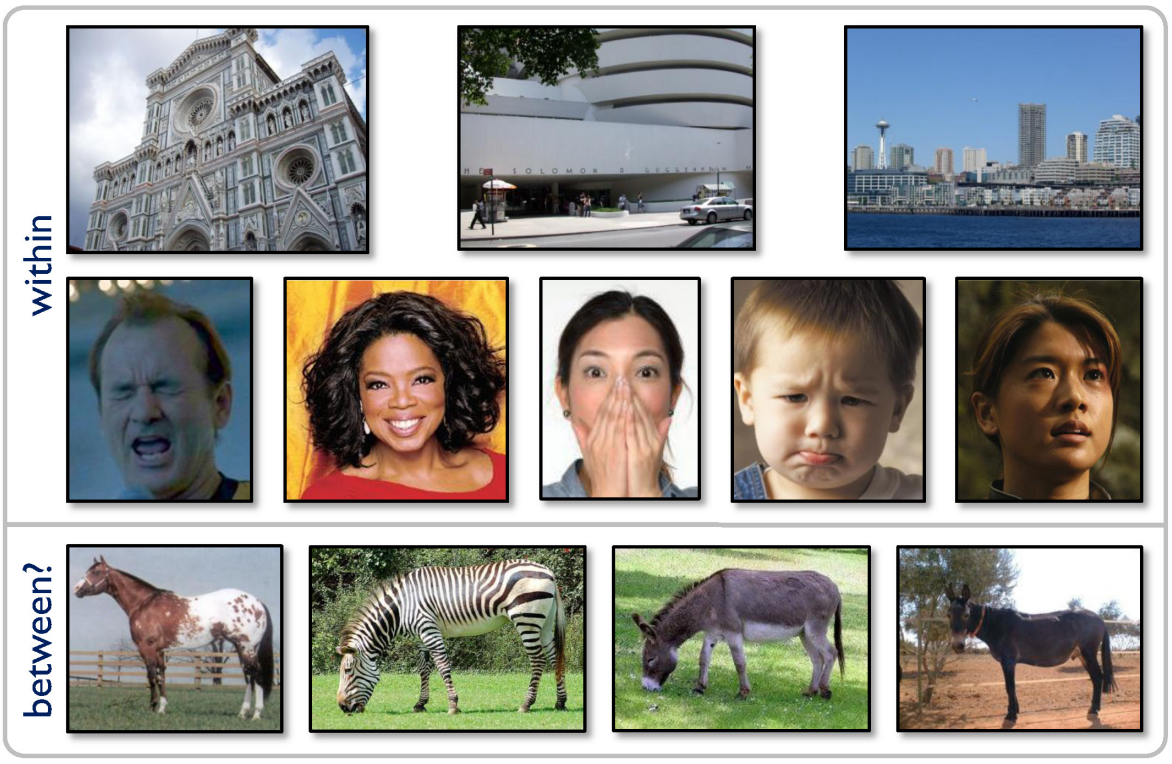

Figure 1.4: **Within-class and between-class variability. Above:** example images of buildings and faces illustrate the wide within-class variability in appearance of objects from particular categories. **Below:** depending on the application, horse, zebra, donkey, and mule could be considered (i) different classes, in which case their visual differences are important, or (ii) the same class (*Equidae*), in which case their visual similarities are important.

We focus on two particular types of features, that are notably different from the sparse interest points and local descriptors in vogue with the computer vision community at present (see Appendix A). Firstly, we investigate the use of fragments of contour (edges), and show that in combination a powerful classifier can be built that is capable of determining the presence or absence of an object in a given region of an image. The second type of feature is based on textural image properties, and is capable of exploiting the appearance, shape (layout), and appearance context of an object. These contour and texture based features are proven on several challenging image datasets, using standard experimental procedure and quantitative measures. Across many different object classes, we obtain very encouraging results, which are in some cases state-of-the-art.

1.5 Contributions

The primary contributions of this thesis are threefold:

- We investigate contour, the outline and interior edges of an object, as a recognition cue. A powerful cue in human visual perception [Biederman & Ju, 1988], we demonstrate that contour is, alone, sufficient for accurate automatic visual recognition.
- We propose two powerful new image feature types. The first of these is a learned codebook of local *contour fragments*, which are matched using a novel formulation of the oriented chamfer distance. The second feature type is texture-based. These features, called *shape filters*, can simultaneously exploit local appearance, approximate shape, and appearance context for accurate and efficient recognition.
- We show how the combination of these two largely orthogonal feature types substantially improves recognition performance above that achieved by either alone.

1.6 Outline

The body of this thesis is divided into five chapters, the first of which is this introduction. Chapter 2 investigates the cue of contour for classification and detection, and presents contour fragments. Chapter 3 introduces shape filters, and shows how they are combined in a conditional random field to give accurate semantic segmentation. Chapter 4 then returns to the tasks of classification and detection, and discusses the combination of contour

fragments with shape filters. We give concluding remarks and discuss limitations and potential future directions in Chapter 5. Finally, Appendix A presents a summary of related work, Appendix B describes the Gentle AdaBoost and Joint Boost algorithms that are used throughout the thesis, and Appendix C illustrates the datasets used in the evaluations of Chapters 2 and 4.

CHAPTER 2

CONTOUR

2.1 Introduction

Consider the images in Figure 2.1, and try to identify the objects present. The object identities are hopefully readily apparent. This simple demonstration confirms the intuition that contour can be used to successfully recognize objects in images, and detailed psychophysical studies such as those of [Biederman & Ju, 1988] bear this out. With this inspiration, we set out to build an automatic object category recognition system that uses only the cue of contour. The most significant contribution of this chapter and its precursor [Shotton *et al.*, 2005] is the demonstration that such a system can accurately recognize objects from challenging and varied object categories. In Chapter 4, we show how to combine several different recognition cues (contour, texture, color, etc.), but for the didactic purposes of this chapter we deliberately throw away color and textural information.

Our system aims to learn, from a small set of training images, a class-specific model for classification and detection in unseen test images. The task of classification is to determine

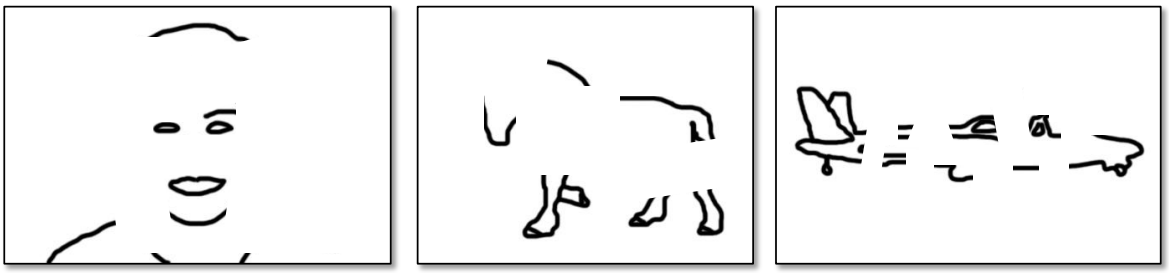

Figure 2.1: **Can you identify the objects from the fragments of contour?** Our innate biological vision system is able to interpret spatially arranged local fragments of contour to recognize the objects present. In this work we show that an automatic computer vision system can also successfully exploit the cue of contour for object recognition. (Object identities are given in Figure 2.3).

the presence or absence of objects of a particular class (category) within an image, answering the question "does this image contain at least one X?", while detection aims to localize any such objects in space and scale, answering "how many Xs are in this image, and where are they?".

We define *contour* as the outline (silhouette) together with the internal edges of the object, while the term *shape* is used to denote the spatial arrangement of object parts. Contour has several advantages over other cues: for example, it is largely invariant to lighting conditions (even silhouetting) and variations in object color and texture, it can efficiently represent image structures with large spatial extents, it varies smoothly with object pose change (up to genus change), and can be matched even in the presence of background clutter. By contrast, image patches and local descriptor vectors tend to match an image less reliably at the boundary, due to interaction with the varying background.

However, the evident power of contour as a recognition cue in nature is somewhat mitigated in computer-based systems by practical realities. Contour is matched against some form of image edge map, but the reliability of general purpose figure-ground segmentation and edge detection is still an area of active research [Boykov & Jolly, 2001; Rother *et al.*, 2004; Dollár *et al.*, 2006]. Indeed the problems of object detection, figure-ground segmentation, and edge detection are intimately bound together: a good segmentation mask gives extremely clean contours, useful for recognition, while localizing the object in scale-space gives an excellent initialization for bottom-up segmentation (see Figure 5.1).

The most significant problem faced by contour-based recognition techniques is that of noisy edge maps and background clutter; the images in Figure 2.3, for example, contain many strong background edges to which the system must be robust. Whole object contours are fairly robust to this clutter, but have poor generalization qualities, and therefore require many exemplars, often arranged hierarchically [Gavrila, 1998], to be useful for detecting deformable objects. Recently, improved models, where whole object templates are divided into parts, have become prominent in computer vision [Fischler & Elschlager, 1973; Burl *et al.*, 1998; Felzenszwalb & Huttenlocher, 2000; Weber *et al.*, 2000; Fergus *et al.*, 2003; Felzenszwalb & Huttenlocher, 2005; Winn & Shotton, 2006]. Parts are individually quite likely to match to background clutter, but in ensemble prove robust and are able to generalize across both rigid and articulated object classes. In Section 2.3 we show how our system learns parts based on contour *fragments* that in combination robustly match both the object outline and repeatable internal edges.

Our preliminary work [Shotton *et al.*, 2005] proved that automatic object recognition was indeed achievable using only contour information. This chapter strengthens and extends that thesis in the following respects:

- A codebook of scale-normalized contour exemplars is learned automatically from the training images (Section 2.2.2), no longer requiring figure-ground segmentation masks for training.
- Recognition is now performed at multiple scales (Section 2.3.2).
- Contour fragments are matched using a new multi-scale formulation of chamfer matching with an explicit penalty for orientation mismatch (Section 2.2.1).
- Object detections are found as the strongest responses of a cascaded sliding-window classifier by the mean shift mode detection algorithm [Comaniciu & Meer, 2002]. We also discuss the probabilistic interpretation of object detection (Section 2.3).
- We demonstrate the effectiveness of a boot-strapping technique which augments the sparse set of training examples used to learn the classifier. This is applied to the training data (Section 2.4.3), and additionally, by assigning a level of trust to the classifier and without compromising procedural integrity, to the test data (Section 2.4.4).
- The evaluation (Section 2.5) is extended to 17 categories, embracing both classification and detection. We introduce a new challenging multi-scale horse dataset, and compare performance with two other contour-based techniques [Opelt *et al.*, 2006c; Ferrari *et al.*, 2006a] and against an interest-point based method [Sivic *et al.*, 2005].

Note that the notation used in this chapter differs somewhat from [Shotton *et al.*, 2005].

A schematic diagram of the algorithm presented in this chapter is shown in Figure 2.2. This references ahead to the relevant sections of this chapter, which is structured as follows.

In Section 2.2, we define contour fragments as sets of oriented edgels (2D points at image edges). A new formulation of chamfer distance, including an explicit cost for orientation mismatch, is explained, and its application to matching at multiple scales is presented. Finally, we discuss how a codebook of robust contour exemplars is learned using a clustering algorithm.

Section 2.3 presents the object detection model. A star constellation of parts arranged about an object centroid is employed, and each part (a contour exemplar) is matched to

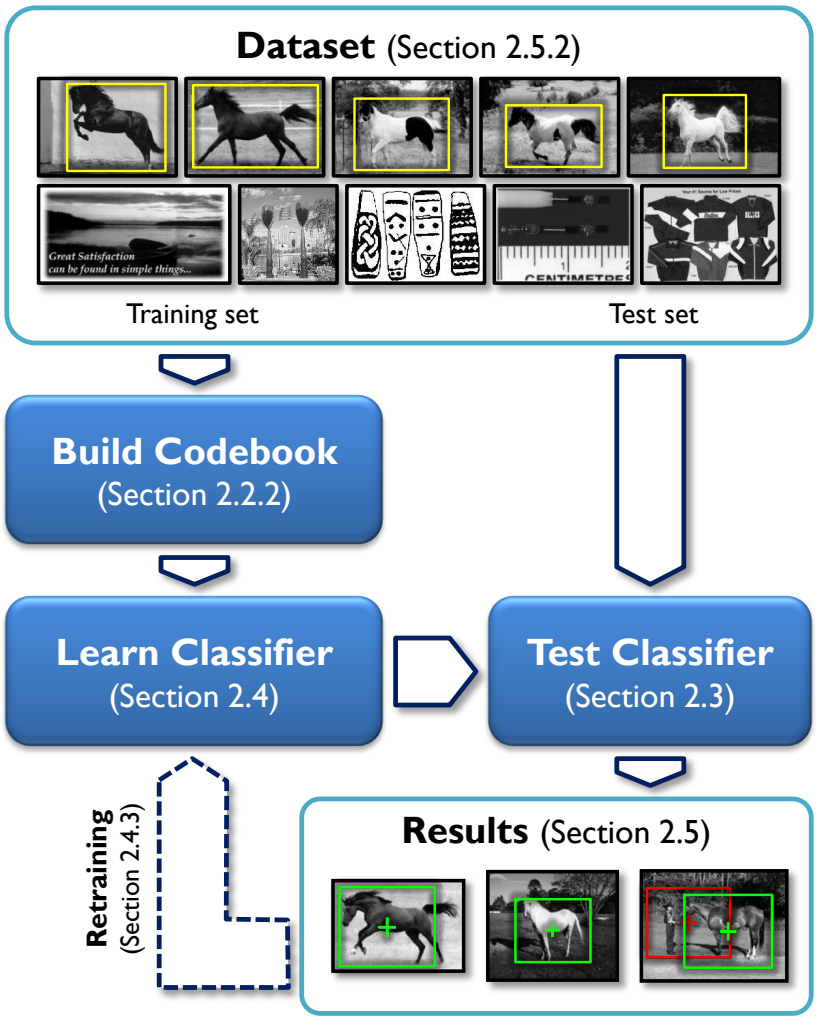

Figure 2.2: **Schema of the contour-based object detection algorithm presented in this chapter.** A set of class images labeled with bounding boxes is merged with a set of background images, and then divided into training and test sets. The training set is used to build a codebook of contour fragments, which are used to construct a classifier for detection. This classifier is evaluated on the test data, and if required, retraining iterates the learning step.

the image edge map using a chamfer distance that incorporates a spatial prior. A boosted classifier infers object presence or absence for centroid hypotheses across scale-space, and mean shift locates a final set of likely detections.

The learning of the classifier is described in Section 2.4. A feature vector of chamfer distances for all exemplars is calculated for a sparse scale-space pattern of training examples, and boosting is used to select discriminative exemplars while learning the classification parameters. A cascade is also learned to speed up the classifier at test time. Lastly, we discuss retraining. Here, an initial classifier is used to identify false positive and false negative detections. The training set is then augmented with examples placed so as to correct these

mistakes, and learning is iterated, resulting in a final classifier with improved generalization properties.

The evaluation is presented in Section 2.5. We use standard classification and detection measures to quantify performance of our technique over several challenging datasets including 17 object classes. The results confirm our hypothesis that contour is a powerful cue for automatic visual recognition, and we demonstrate excellent results for both rigid and articulated classes (see Figures 2.12 and 2.15). Our comparisons with other contour-based techniques, and with a method that uses sparse local descriptors, show strong, state-of-the-art recognition performance.

Finally, Section 2.6 concludes this chapter with a summary of our findings.

Related Work

First, however, we briefly discuss other techniques that use contour fragments. Broader references, including those to methods that match whole contour templates, are presented in Appendix A.

[Fergus *et al.*, 2004] augmented the constellation model with contour fragment features, but their technique only exploits fairly clean, planar curves with at least two points of inflection. In [Kumar *et al.*, 2004], contour fragments learned from video sequences were arranged in Pictorial Structures [Fischler & Elschlager, 1973; Felzenszwalb & Huttenlocher, 2000, 2005] and used for detection of articulated objects; good results were obtained, although tracking of video sequences or manual labeling of parts was needed for learning. [Borenstein *et al.*, 2004] used image and contour fragments for segmentation, but did not address classification or detection. A similar technique to [Shotton *et al.*, 2005] was proposed in [Opelt *et al.*, 2006a].

Other methods have also used local contour descriptors. Rigid objects were addressed effectively in [Mikolajczyk *et al.*, 2003]. Shape contexts [Belongie *et al.*, 2002] describe sampled edge points in a log-polar histogram. The geometric blur descriptor was used in [Berg *et al.*, 2005] to match deformable objects between pairs of images. Most recently, [Ferrari *et al.*, 2006a] combined groups of adjacent segments of contour into invariant descriptors, and the use of sliding windows of localized histograms enabled object detection.

Figure 2.3: **Answers for the recognition question posed in Figure 2.1.**

2.2 Contour Fragments

In this section we discuss contour fragments, giving their precise definition and detailing how they are extracted from image edge maps and clustered into a class-specific codebook of exemplars (Section 2.2.2). First however, we present our new formulation of chamfer matching with an explicit penalty for mismatch in orientation.

2.2.1 Chamfer Matching

The chamfer distance function, originally proposed in [Barrow *et al.*, 1977], measures the similarity of two contours at a certain relative location. It is a smooth measure with considerable tolerance to noise and misalignment in position, scale and rotation, and hence very suitable for matching our locally rigid contour fragments to noisy edge maps. It has already proven capable of and efficient at recognizing whole object outlines (e.g. [Gavrila, 1998; Stenger *et al.*, 2003; Leibe *et al.*, 2005]), and here we extend it for use in a multi-scale parts-based categorical recognition model.

In its most basic form, chamfer distance (for 2D relative translation $\mathbf{x}$) takes two sets of edgels, a template T and an edge map E, and evaluates the (asymmetric) distance as:

$$d_{\text{cham}}^{(T,E)}(\mathbf{x}) = \frac{1}{|T|} \sum_{\mathbf{x}_\text{t} \in T} \min_{\mathbf{x}_\text{e} \in E} \|(\mathbf{x}_\text{t} + \mathbf{x}) - \mathbf{x}_\text{e}\|_2 \ , \tag{2.1}$$

where $|T|$ denotes the number of edgels in template T, and $\| \cdot \|_2$ denotes the l_2 norm. The chamfer distance therefore gives the mean distance of edgels in T to their closest edgels in E. For clarity of presentation, we omit the superscript (T, E) below where possible.

The chamfer distance can be efficiently computed via the *distance transform* (DT) of E,

DT_E. This is an image in which each pixel gives the distance to the closest edgel in E:

$$\mathrm{DT}_E(\mathbf{x}) = \min_{\mathbf{x}_\mathrm{e} \in E} \|\mathbf{x} - \mathbf{x}_\mathrm{e}\|_2 \ . \tag{2.2}$$

Hence the $\min$ operation in (2.1) becomes a simple look-up such that $d_\mathrm{cham}(\mathbf{x})$ can be computed as:

$$d_\mathrm{cham}(\mathbf{x}) = \frac{1}{|T|} \sum_{\mathbf{x}_\mathrm{t} \in T} \mathrm{DT}_E(\mathbf{x}_\mathrm{t} + \mathbf{x}) \ . \tag{2.3}$$

We also compute the *argument* distance transform (ADT) which gives the locations of the closest points in E:

$$\mathrm{ADT}_E(\mathbf{x}) = \arg\min_{\mathbf{x}_\mathrm{e} \in E} \|\mathbf{x} - \mathbf{x}_\mathrm{e}\|_2 \ . \tag{2.4}$$

The exact Euclidean DT and ADT can be computed simultaneously in linear time using the algorithm of [Felzenszwalb & Huttenlocher, 2004].

It is standard practice to truncate the distance transform to a value τ:

$$\mathrm{DT}_E^\tau(\mathbf{x}) = \min(\mathrm{DT}_E(\mathbf{x}), \tau) \ , \tag{2.5}$$

which adds robustness to the basic chamfer distance by ensuring that missing edgels due to noisy edge detection do not have too severe an effect. Additionally it allows the chamfer distance to be normalized to a standard range $[0, 1]$:

$$d_{\mathrm{cham},\tau}(\mathbf{x}) = \frac{1}{\tau|T|} \sum_{\mathbf{x}_\mathrm{t} \in T} \mathrm{DT}_E^\tau(\mathbf{x}_\mathrm{t} + \mathbf{x}) \ . \tag{2.6}$$

Edge Orientation

A further, much greater improvement than truncation by τ is given by exploiting edge orientation information in the form of edge gradients. This orientation cue alleviates problems caused by background clutter edgels since they are unlikely to align in both orientation and position. One popular extension to basic chamfer matching is to divide the edge map and template into discrete orientation channels and sum the individual chamfer scores [Olson & Huttenlocher, 1997; Stenger *et al.*, 2003]. However, it is not clear how many channels to use, nor how to avoid artifacts at the channel boundaries.

Instead, we augment the robust chamfer distance (2.6) with a continuous and explicit cost for orientation mismatch, given by the mean difference in orientation between edgels

in template T and the nearest edgels in edge map E:

$$d_{\text{orient}}(\mathbf{x}) = \frac{2}{\pi|T|} \sum_{\mathbf{x}_t \in T} |\phi(\mathbf{x}_t) - \phi(\text{ADT}_E(\mathbf{x}_t + \mathbf{x}))| \ . \tag{2.7}$$

The function $\phi(\mathbf{x})$ gives the orientation of edgel $\mathbf{x}$ modulo π, and $|\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)|$ gives the smallest circular difference between $\phi(\mathbf{x}_1)$ and $\phi(\mathbf{x}_2)$. Edgels are taken modulo π because, for edgels on the outline of an object, the sign of the edgel gradient is not a reliable signal since it depends on the intensity of the background. The normalization by $\frac{\pi}{2}$ ensures that $d_{\text{orient}}(\mathbf{x}) \in [0, 1]$, since $|\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)| < \frac{\pi}{2}$.

Our final improved distance function, which we call the *oriented chamfer distance*, is then a simple linear interpolation between the distance and orientation terms

$$d_\lambda(\mathbf{x}) = (1 - \lambda) \cdot d_{\text{cham},\tau}(\mathbf{x}) + \lambda \cdot d_{\text{orient}}(\mathbf{x}) \ , \tag{2.8}$$

where the *orientation specificity* parameter λ weights the distance and orientation terms. As we shall see below, λ is learned for each contour fragment separately, giving improved discrimination power compared with a shared, constant λ. The distance and orientation terms in (2.8) are illustrated in Figure 2.4. Note that oriented chamfer matching is considerably more storage efficient than using discrete orientation channels. The precise mathematical form of the oriented chamfer distance (2.8) has been clarified slightly since our original formulation in [Shotton *et al.*, 2005].

In Section 2.5.3 and Figure 2.10, we compare the performance of our oriented chamfer distance against 8-channel chamfer matching and Hausdorff matching [Huttenlocher & Rucklidge, 1992]. The Hausdorff distance function is essentially the basic chamfer distance (2.1), but the summation is replaced by a maximization. We show that our continuous use of orientation information, with the ability to learn per-part orientation specificities, provides a considerable improvement over both these methods.

Matching at Multiple Scales

We extend oriented chamfer matching to multiple scales. This extension proves to be especially simple because, rather than using an image pyramid [Borgefors, 1988], we rescale the templates T, keeping the size of edge map E constant. We thus redefine template T to be a set of *scale-normalized* edgels. To compute the chamfer distance at scale s between T and

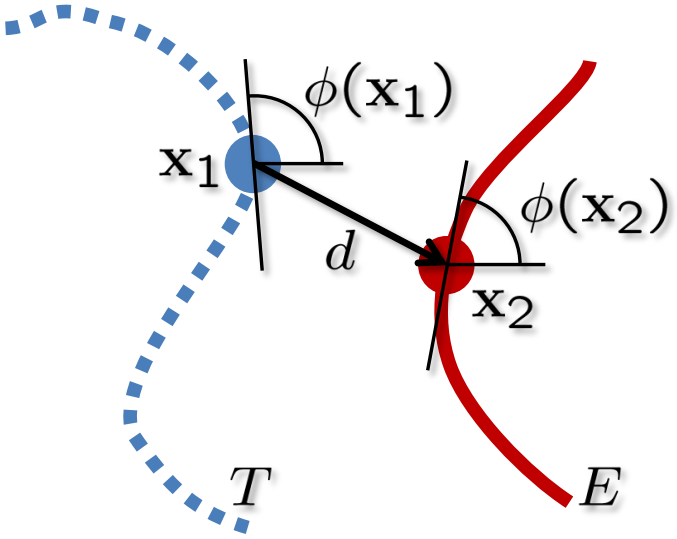

Figure 2.4: **Oriented chamfer matching.** For edgel $\mathbf{x}_1$ (blue circle) in template T (dotted blue curve), the contribution to the oriented chamfer distance is determined by the distance d from $\mathbf{x}_1$ to the nearest edgel $\mathbf{x}_2$ (red circle) in edge map E (solid red curve), and the difference between the edgel gradients at these points, $|\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)|$.

the (original, unscaled) edge map E, we use the scaled edgel set $sT = \{s\mathbf{x}_t \text{ s.t. } \mathbf{x}_t \in T\}$ and calculate:

$$d_\lambda^{(T,E)}(\mathbf{x}, s) = d_\lambda^{(sT,E)}(\mathbf{x}) \; . \tag{2.9}$$

We round the scaled edgel positions in sT to the nearest integer; alternatively one could interpolate the distance transform.

At smaller scales the edgels in template T are squashed closer together and some may even alias to the same location in the distance transform, while at larger scales, the edgels are stretched apart with gaps forming between them. However, due to the normalization by $|T|$, the chamfer distances can reasonably be compared across scales, as required for a scale-invariant model.

Approximate Chamfer Matching

For efficiency, one does not need to perform the complete sums over template edgels in (2.6) and (2.7). Each sum represents an empirical average, and so one can sum over only a fraction of the edgels, adjusting the normalization accordingly. This provides a good approximation to the true chamfer distance function in considerably reduced time. In practice, even matching only 20% of edgels gave no decrease in detection performance, as demonstrated in Section 2.5.5.

2.2.2 Building a Fragment Codebook

To use contour fragments for object recognition, we must first come up with a set of representative fragments. In selecting these, one has a choice in their specificity. One could use completely generic fragments such as lines, corners, and T-junctions and hope that in combination they can be made discriminative [Ferrari *et al.*, 2006a]. Instead, we create a *class-specific* set of fragments so that, for instance, the class horse will give rise to fragments corresponding to regions we know to be head, back, and forelegs, among others, as illustrated in Figure 2.7. Even individually, these fragments can be indicative of object presence in an image, and in combination prove very powerful for object detection, as we shall demonstrate.

In our earlier work [Shotton *et al.*, 2005], we built a codebook by extracting clean fragments of contour from ground truth segmentations of the training data. However, hand-segmenting a large set of images is somewhat laborious, so in this work we present an improved formulation that does not require segmentations, only bounding boxes around the training objects.

The outline of our codebook learning algorithm is as follows. We start with a large, randomly chosen initial set of fragments, which is clustered based on a symmetrized chamfer distance (see ahead to (2.11)). Around 10000 fragments are clustered to about 500 clusters. Each cluster is subdivided to find fragments that agree in centroid position. The resulting sub-clusters form the final codebook of fragments. We also refer to these codebook fragments as contour *exemplars*. We show in Section 2.5.6 that using the new learned codebook from unsegmented images can be even more powerful than a codebook learned from segmented images.

Contour fragments are extracted from edge maps computed using the Canny edge detector [Canny, 1986], although at first no thresholding is applied, and hysteresis is not used. Each training image contains a number of objects labeled with bounding boxes. Bounding box $b = (\mathbf{b}_{\mathrm{tl}}, \mathbf{b}_{\mathrm{br}})$ implicitly defines an object *centroid* $\mathbf{x} = \frac{1}{2}(\mathbf{b}_{\mathrm{tl}} + \mathbf{b}_{\mathrm{br}})$, and an object *scale* $s = \sqrt{\mathrm{area}(b)}$. The centroid and scale are illustrated in Figure 2.8. Object scales are only used in ratios, and so their absolute values are not significant.

The initial set of contour fragments is generated as follows. A training object and a rectangle $r = (\mathbf{r}_{\mathrm{tl}}, \mathbf{r}_{\mathrm{br}})$ enclosed within the bounding box of the object are chosen, both uniformly at random. We define vector $\mathbf{x}_{\mathrm{f}} = \frac{1}{s}(\mathbf{r}_{\mathrm{cen}} - \mathbf{x})$ as the (scale-normalized) vector

from the object centroid $\mathbf{x}$ to the rectangle center $\mathbf{r}_{\text{cen}} = \frac{1}{2}(\mathbf{r}_{\text{tl}} + \mathbf{r}_{\text{br}})$. Let $E_r = \{\mathbf{x}_{\text{r}}\}$ denote the set of absolute image positions of edgels within rectangle r. The set of scale-normalized fragment edgels is:

$$T = \left\{ \frac{1}{s}(\mathbf{x}_{\text{r}} - \mathbf{r}_{\text{cen}}) \text{ s.t. } \mathbf{x}_{\text{r}} \in E_r \right\} . \tag{2.10}$$

To reduce the number of empty and overly generic fragments such as small straight lines, fragments with edgel density $\frac{|E_r|}{\text{area}(r)}$ below a threshold η_1 are immediately discarded. Fragments with edgel density above a threshold η_2 are also discarded, since these are likely to contain many background clutter edgels and even if not, will be expensive to match. Edgel sets E_r are computed as $E_r = \{\mathbf{x} \in C \text{ s.t. } \mathbf{x} \in r \text{ and } \|\boldsymbol{\nabla} I\|_{\mathbf{x}} > t\}$. This equation uses the image gradient $\|\boldsymbol{\nabla} I\|$ at the set of edge points C, given by the Canny non-maximal suppression algorithm (see examples in Figure 2.14). Rather than fix an arbitrary threshold t, we choose a random t for each fragment (uniformly, within the central 50% of the range $[\min_{\mathbf{x}} \|\boldsymbol{\nabla} I\|_{\mathbf{x}}, \max_{\mathbf{x}} \|\boldsymbol{\nabla} I\|_{\mathbf{x}}]$), so that at least some of the initial fragments are relatively clutter-free. As we shall see shortly, the clustering step can then pick out these cleaner fragments to use as exemplars.

Finally, to ensure the initial set of contour fragments covers the possible appearances of an object, a small uniformly random transformation is applied to each fragment: a scaling $\log s \in [-\log s_{\text{rnd}}, \log s_{\text{rnd}}]$ and rotation $\theta \in [-\theta_{\text{rnd}}, \theta_{\text{rnd}}]$ about the fragment center is applied to the edgels, and the vector $\mathbf{x}_{\text{f}}$ is translated (by $x \in [-t_{\text{rnd}}, t_{\text{rnd}}]$ and $y \in [-t_{\text{rnd}}, t_{\text{rnd}}]$) and rotated (by $\phi \in [-\phi_{\text{rnd}}, \phi_{\text{rnd}}]$) about the object centroid. Several differently perturbed but otherwise similar fragments are likely to result, given the large number of fragments extracted.

Fragment Clustering

Figure 2.5 shows example fragments extracted at random, for both segmented and unsegmented training images. Clearly, the fragments from unsegmented images are fairly noisy, though some are less cluttered than others. A clustering step is therefore employed with the intuition that the resulting exemplars (cluster centers) are likely to be relatively clean and clutter free.

To this end we compare all pairs (T_i, T_j) of fragments in the initial set. This is done in a symmetric fashion as follows:

$$d_{i,j} = d_{\lambda}^{(s_j T_i, s_j T_j)}(\mathbf{0}) + d_{\lambda}^{(s_i T_j, s_i T_i)}(\mathbf{0}) , \tag{2.11}$$

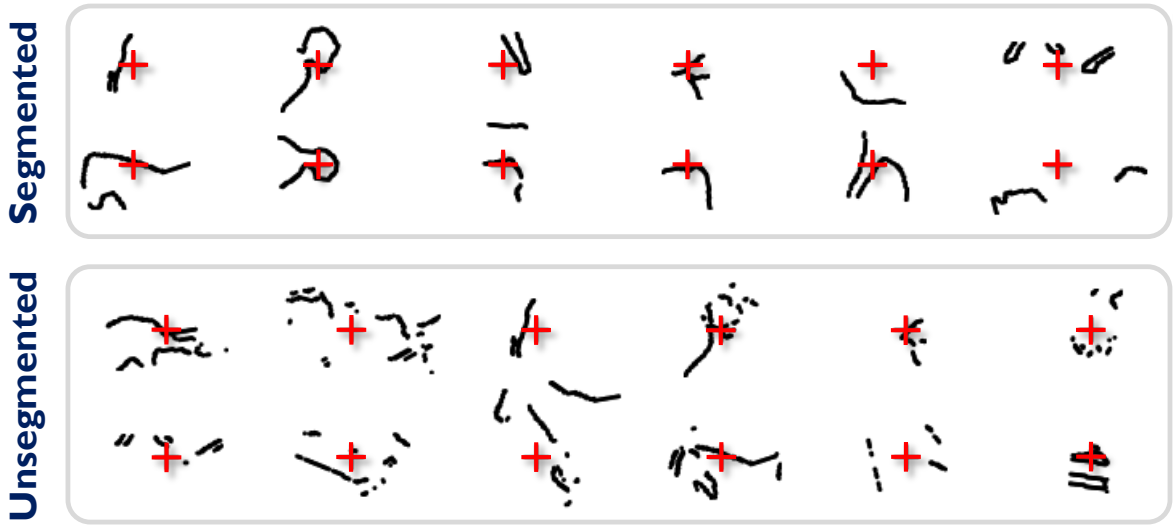

Figure 2.5: **Initial set of contour fragments.** Examples of contour fragments extracted at random from horse images. The red crosses represent the origins of the fragments, i.e. the vectors $(0,0)^T$ in the coordinate systems implicitly defined in (2.10). **Top:** clean fragments can be extracted from segmented training images. **Bottom:** much noisier fragments tend to be extracted when segmentations are not provided.

so that the fragments are scaled (first both to s_j, then both to s_i) and compared at zero relative offset. Clustering is performed on the matrix $d_{i,j}$ using the k-medoids algorithm, the analogue of k-means for non-metric spaces. For the purposes of clustering, a (single) value for λ is used. This was chosen to maximize the difference between histograms of distances $d_{i,j}$ for within-cluster and between-cluster fragment pairs.

Example fragment clusters are shown in Figure 2.6. As hoped, clusters contain contour fragments of similar appearance, and even for unsegmented ground truth, the cluster centers tend to be clean contour fragments. However, this purely appearance-based clustering does not take the vectors $\mathbf{x}_\mathrm{f}$ from the object centroid into account. We desire each contour fragment to give a unique and reliable estimate of the object centroid, and so we split each cluster into sub-clusters which agree on $\mathbf{x}_\mathrm{f}$, as follows. Each fragment casts a vote for the object centroid, and modes in the voting space are found using mean shift mode estimation [Comaniciu & Meer, 2002]. Each mode defines a sub-cluster, with all fragments within a certain radius of the mode of $\mathbf{x}_\mathrm{f}$ assigned to that sub-cluster. To ensure high quality sub-clusters, only those with a sufficient number of fragments are kept (in all experiments in this chapter, five fragments were required). The sub-clustering procedure is iterated for those fragments not assigned to a sub-cluster, until no new sub-clusters are generated.

Contour fragments within each sub-cluster now agree both in appearance, in terms of

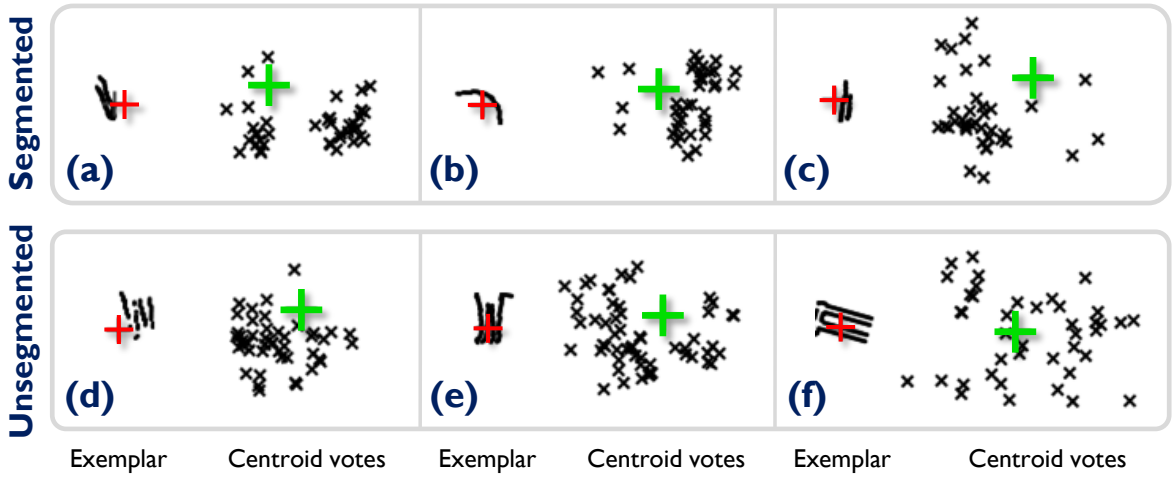

Figure 2.6: **Fragment clustering. Top:** example clusters from segmented images. **Bottom:** example clusters from unsegmented images. In each example, shown left is the cluster center (exemplar), and right are the locations $\mathbf{x}_f$ (black diagonal crosses) of all the cluster member fragments, relative to the object centroid (green cross). Note that (i) appearance-only clustering can give clusters with multiple modes in the voting space (e.g. **(a)** and **(b)**), (ii) segmented fragments tend to cluster more cleanly than unsegmented fragments, and (iii) the locations of background cluster members (e.g. **(f)**) are scattered widely with no discernible pattern.

low mutual chamfer distances (2.11), and also location relative to the object centroid. Within each sub-cluster, the center fragment (the fragment $\bar{T}$ with lowest average distance to other fragments) is used to form an exemplar $F = (\bar{T}, \bar{\mathbf{x}}_f, \sigma)$, where $\bar{\mathbf{x}}_f$ and σ are respectively the scale-normalized mean and radial variance of the centroid vectors $\mathbf{x}_f$. The exemplars from all sub-clusters are combined to form the codebook $\mathcal{F} = \{F\}$. Figure 2.7 illustrates the sub-clustered contour fragments. Note that the final exemplars are very class specific since random background fragments are highly unlikely to repeatably agree in position as well as appearance. Our clustering algorithm has also been able to obtain clean contour exemplars from unsegmented images.

The clustering step is somewhat similar to that used in [Leibe & Schiele, 2003], except that (i) we cluster contour fragments rather than image patches, and (ii) each resulting sub-cluster has a particular location relative to the centroid as opposed to having the multiple centroid votes of [Leibe & Schiele, 2003].

2.3 Object Detection

In this section we describe how contour exemplars are combined in a parts-based object detection model. Parts are matched to an image edge map using the scale-invariant oriented

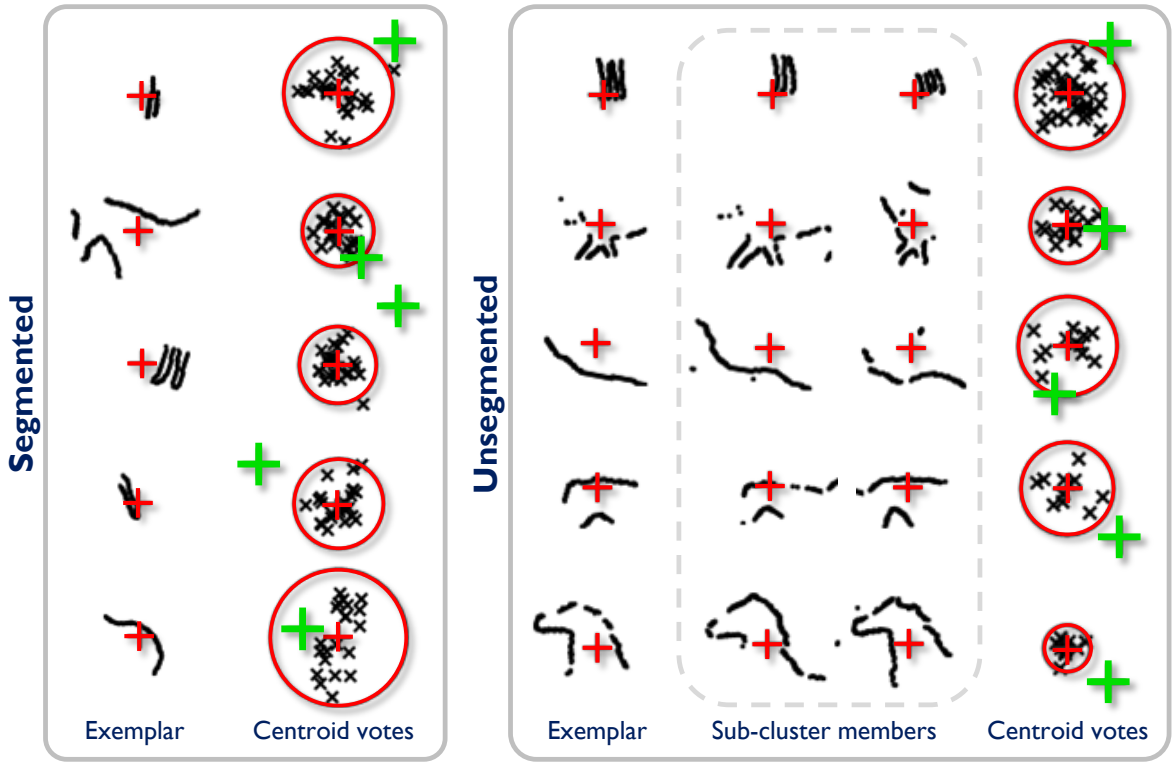

Figure 2.7: **Example contour exemplars. Left:** exemplars from segmented horse images. **Right:** exemplars from unsegmented images. Each row represents a sub-cluster containing contour fragments that agree on centroid location $\mathbf{x}_f$ as well as in appearance, in terms of low mutual chamfer distances (2.11). Within each row are shown the contour exemplar (the center of the sub-cluster), example sub-cluster members, and the locations $\mathbf{x}_f$ (black diagonal crosses) of the sub-cluster member fragments relative to the object centroid (large green cross). The red circle is centered on a small red cross, and these indicate, respectively, the radial uncertainty σ, and the mean $\bar{\mathbf{x}}_f$ of votes relative to the centroid. Note that (i) we obtain representative, class specific fragments of contour to use as exemplars for recognition, (ii) through clustering we get clean contour exemplars even without segmented training data, and (iii) we obtain an accurate estimate of location and location uncertainty relative to the object centroid.

chamfer distance with priors on their spatial layout, and combined in a boosted classifier. The classifier is evaluated across the scale-space of the image, and mean shift mode detection produces a final set of confidence-valued object detections. The only image information used by the detector is the edge map E, which is computed using the Canny edge detector [Canny, 1986] (although no hysteresis is applied).

2.3.1 Parts-Based Object Model

Most modern categorical object recognition systems such as [Agarwal & Roth, 2002; Fergus *et al.*, 2003; Felzenszwalb & Huttenlocher, 2005] attempt to recognize an object as the sum of its parts, rather than the object as a whole. This gives numerous advantages, allowing recognition of partially occluded objects [Winn & Shotton, 2006], and significantly improving efficiency and accuracy while decreasing training data requirements when modeling classes with considerable articulation and within-class variation (different individuals, body configurations, facial expressions). Some existing systems, e.g. [Fergus *et al.*, 2003], are computationally limited to a small number of parts, but our technique can efficiently cope with larger numbers, of the order of 100. The resulting over-complete model has built-in redundancy with tolerance to within-class variation and different imaging conditions such as lighting, occlusion, clutter, and small pose changes.

The spatial layout of parts is clearly informative, although the degree to which it is modeled varies enormously. One popular and remarkably successful technique, the *bag-of-words* model [Sivic & Zisserman, 2003; Csurka *et al.*, 2004; Sivic *et al.*, 2005; Fergus *et al.*, 2005] throws away all spatial information and exploits the repeatable co-occurrence of features to recognize objects or scenes. At the opposite extreme, for a small number of parts, a full joint spatial layout distribution can be learned [Fergus *et al.*, 2003].

Our algorithm lies between these two extremes, using a star shaped constellation illustrated in Figure 2.8, where the locations of the parts are constrained through a single fiducial point on the object, the centroid. A part $P = (F, \lambda, \theta, a, b)$ in our model is a contour exemplar $F = (\bar{T}, \bar{\mathbf{x}}_\mathrm{f}, \sigma)$ paired with several learned parameters: λ is the orientation specificity of the part in (2.8), while θ thresholds, and a and b confidence-weight the part detections, as described below.

For an object centroid hypothesis with location $\mathbf{x}$ and scale s, part P is expected to match the image edge map E near position $\mathbf{x} + s\bar{\mathbf{x}}_\mathrm{f}$, with spatial uncertainty $s\sigma$. The chamfer distance is therefore weighted with a cost increasing away from the expected position. Finding

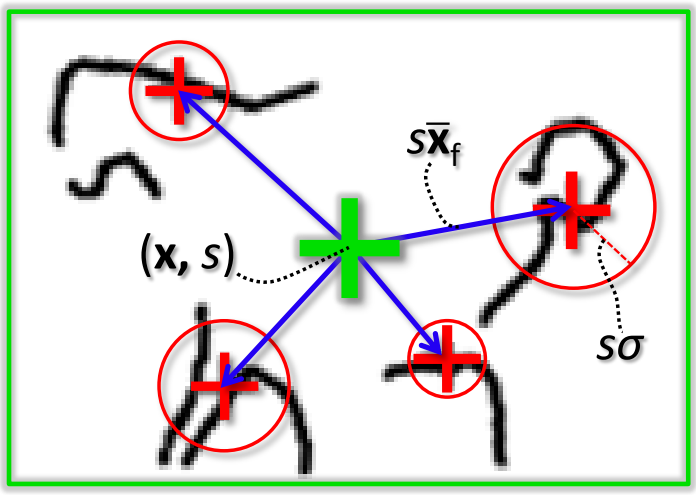

Figure 2.8: **Object star constellation.** Object parts (black fragments of contour) are located about the object centroid at $(\mathbf{x}, s)$ (green cross). The object bounding box is shown in green. Each part has a spatial location $\bar{\mathbf{x}}_f$ relative to the centroid (blue arrow) and a spatial uncertainty σ (red circle), both learned when constructing the fragment codebook. Both the spatial location and uncertainty are scaled by object scale s. For clarity of presentation, only four parts are shown here; in practice, about 100 parts are used.

the minimum weighted distance thus allows a degree of spatial flexibility in matching. The location of this minimum is given by

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}'} \left(d_\lambda^{(\bar{T},E)}(\mathbf{x}', s) + w_{s\sigma}(\|\mathbf{x}' - (\mathbf{x} + s\bar{\mathbf{x}}_f)\|_2) \right) , \tag{2.12}$$

where $w_\sigma(x)$ is the radially symmetric spatial weighting function for which we use the quadratic

$$w_\sigma(x) = \begin{cases} \frac{x^2}{\sigma^2} & \text{if } |x| \leq \sigma \\ \infty & \text{otherwise.} \end{cases} \tag{2.13}$$

The *part response* v for centroid hypothesis $(\mathbf{x}, s)$ is defined as the chamfer distance at the best match $\mathbf{x}^\star$

$$v_{[F,\lambda]}(\mathbf{x}, s) = d_\lambda^{(\bar{T},E)}(\mathbf{x}^\star, s) , \tag{2.14}$$

and this is used in the classifier (2.15) described below.

The most efficient method of finding the minimum in (2.12) depends on the density of candidate centroids $\mathbf{x}$. With candidates very close together it is best to use the algorithm of [Felzenszwalb & Huttenlocher, 2004], which can compute $\mathbf{x}^\star$ over the whole image at once (since w is a convex function). For our purposes however, we found it more efficient to perform a brute force search where required, as the candidate centroids were sufficiently

far apart.

2.3.2 Detecting Objects

Sliding window classification [Viola & Jones, 2001; Agarwal & Roth, 2002; Ferrari *et al.*, 2006a] is a simple, effective technique for object detection. A probability $P(\text{obj}_{(\mathbf{x},s)})$ of object presence at location $(\mathbf{x}, s)$ is calculated across scale-space using a boosted classifier which combines multiple part responses v from (2.14). These probabilities are far from independent, since the presence of two distinct neighboring detections is highly unlikely, for example. Hence a non-maximal suppression step, for which we employ mean shift mode estimation [Comaniciu & Meer, 2002], is used to select local maxima as the final set of detections.

One must choose a set $\mathcal{X}$ of centroid scale-space location hypotheses, sampled frequently enough to allow detection of all objects present, but sparsely enough to avoid undue computational overhead. A fixed number of scales is chosen, equally spaced logarithmically to cover the range of scales in the training data, which we assume is representative. Space is sampled over a regular grid with spacing $s\Delta_{\text{grid}}$ for constant Δ_{grid} (optimized by hand against the validation set). By increasing the spacing with scale, we can safely improve efficiency due to the greater tolerance to misalignment given by the enlarged search window in (2.12).

Classifier

We employ a boosted classifier to compute probabilities $P(\text{obj}_{(\mathbf{x},s)})$. This combines the part responses v (2.14) for parts $P_1, \ldots, P_M$ in an additive model of the form

$$H(\mathbf{x}, s) = \sum_{m=1}^{M} h_m(\mathbf{x}, s) = \sum_{m=1}^{M} a_m[v_{[F_m,\lambda_m]}(\mathbf{x}, s) > \theta_m] + b_m \ , \tag{2.15}$$

with the binary indicator function $[\textit{condition}] = 1$ if *condition* is true, 0 otherwise. Each *weak learner* h_m (corresponding to part P_m in the model) is a decision stump which assigns a weak confidence value (in the range $-\infty$ to $+\infty$) according to the comparison of part response $v_{[F_m,\lambda_m]}$ to threshold θ_m. The weak learner confidences are summed to produce a strong hypothesis confidence H, which can then be interpreted as a probability using the logistic transformation [Friedman *et al.*, 2000]:

$$P(\text{obj}_{(\mathbf{x},s)}) = \frac{1}{1 + \exp(-H(\mathbf{x}, s))} \ . \tag{2.16}$$

Mode Detection

Evaluating $P(\text{obj}_{(\mathbf{x},s)})$ for all scale-space centroid hypotheses $(\mathbf{x}, s) \in \mathcal{X}$ is the starting point for classification and detection. We can write the classification task as that of estimating

$$P(\bigcup_{(\mathbf{x},s)\in\mathcal{X}} \text{obj}_{(\mathbf{x},s)}) \;=\; 1 - P(\bigcap_{(\mathbf{x},s)\in\mathcal{X}} \overline{\text{obj}}_{(\mathbf{x},s)}) \,, \tag{2.17}$$

and the detection task as finding the set of detections $\mathcal{D}$ that maximizes

$$P(\bigcap_{(\mathbf{x},s)\in\mathcal{D}} \text{obj}_{(\mathbf{x},s)} \bigcap_{(\mathbf{x},s)\in\mathcal{X}\setminus\mathcal{D}} \overline{\text{obj}}_{(\mathbf{x},s)}) \,, \tag{2.18}$$

where $\overline{\text{obj}}_{(\mathbf{x},s)}$ represents the event that an object is not present at $(\mathbf{x}, s)$.

Unfortunately, the posteriors of neighboring windows $(\mathbf{x}_1, s_1)$ and $(\mathbf{x}_2, s_2)$ cannot be treated as independent, in other words

$$P(\text{obj}_{(\mathbf{x}_1,s_1)}, \text{obj}_{(\mathbf{x}_2,s_2)}) \neq P(\text{obj}_{(\mathbf{x}_1,s_1)})P(\text{obj}_{(\mathbf{x}_2,s_2)}) \,, \tag{2.19}$$

and this greatly complicates the matter of computing (2.17) and (2.18). Finding a method to calculate or approximate the classification and detection probabilities in a principled manner is a very hard challenge, and one that the authors do not believe the community has yet satisfactorily addressed. We leave this challenge, beyond the scope of this work, for future endeavors; one possible solution could incorporate a Markov random field [Geman & Geman, 1984] prior over the hypotheses, disallowing overlapping detections so that the remaining detections are truly independent.

Instead, we use the powerful and now fairly standard technique of mean shift mode estimation [Comaniciu & Meer, 2002] on the hypothesized locations $(\mathbf{x}, s) \in \mathcal{X}$ weighted by their scaled posterior probabilities $s^2 P(\text{obj}_{(\mathbf{x},s)})$, similarly to [Leibe & Schiele, 2004]. Multiplying by s^2 compensates for the proportionally less dense hypotheses at larger scales. The algorithm models the non-parametric distribution over the hypothesis space with the kernel density estimator

$$P(\mathbf{x}, s) \propto \sum_{(\mathbf{x}_i,s_i)\in\mathcal{X}} s_i^2 P(\text{obj}_{(\mathbf{x}_i,s_i)}) K\left(\frac{x - x_i}{h_\text{x}}, \frac{y - y_i}{h_\text{y}}, \frac{\log s - \log s_i}{h_\text{s}}\right) \,, \tag{2.20}$$

where $\mathbf{x} = (x, y)^T$, the Gaussian kernel K uses bandwidths h_x, h_y and h_s for the x, y, and

scale dimensions respectively, and the scale dimension is linearized by taking logarithms. The mean shift mode estimation procedure efficiently locates modes (local maxima) of the distribution, which are used as the final set of detections. The density estimate at each mode is used as a confidence value for the detection. To get a confidence value for image classification, we simply take the density estimate at the global maximum.

2.4 Learning

We describe in this section how the set of parts $\mathcal{P}$ is learned from the contour exemplars $\mathcal{F}$. Recall that a part $P = (F, \lambda, \theta, a, b)$ consists of a contour exemplar $F \in \mathcal{F}$ and parameters λ, θ, a and b. The challenge of learning is therefore to select discriminative exemplars F from the codebook and learn the parameters of the classifier (2.15). We describe the boosting algorithm as applied to our problem, discuss how a cascade can be learned to improve test speed, and finally how retraining on both the training and test sets can improve detection accuracy.

2.4.1 Boosting

We employ the Gentle AdaBoost algorithm [Friedman *et al.*, 2000], detailed in Appendix B, to learn the classifier in (2.15). The algorithm takes as input a set of training examples i each consisting of feature vector $\mathbf{f}_i$ paired with target value $z_i = \pm 1$, and iteratively builds a classifier which should generalize to new data.

For our purposes, training example i represents location $(\mathbf{x}_i, s_i)$ in one of the training images. The target value z_i specifies the presence ($z_i = +1$) or absence ($z_i = -1$) of the object class. The feature vector $\mathbf{f}_i$ contains the responses $v_{[F,\lambda]}(\mathbf{x}_i, s_i)$ (2.14) for all contour exemplars $F \in \mathcal{F}$, and all orientation specificities λ from a discrete set Λ. A given dimension d in the feature vector therefore encodes a pair (F, λ), and, since decision stumps are used [Torralba *et al.*, 2007], each learned weak learner directly corresponds to an object model part P.

We are free to choose the number, locations, and target values of the training examples. One could densely sample each training image, computing feature vectors for examples at every point on a grid in scale-space. This is however unnecessarily inefficient because the minimization over $\mathbf{x}'$ in (2.12) means that neighboring locations often have near identical feature vectors.

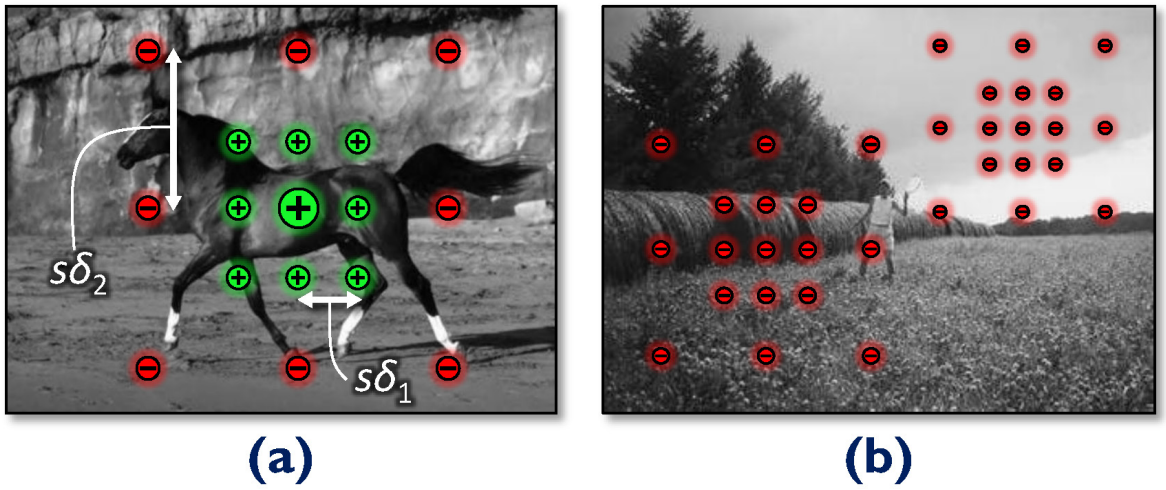

Figure 2.9: **Training examples. (a)** A pattern of positive (green ⊕) and negative (red ⊖) examples are arranged about the true object centroid (the central, larger green ⊕). The positive and negative examples are spaced on a grid of size δ_1 and δ_2 respectively, scaled by the object scale s. At each example, a feature vector of part responses is computed and passed to the boosting algorithm. **(b)** For images with no objects present (background images), copies of the same pattern as before (though now with all negative examples) are placed at a number of random scale-space locations. Note that the patterns of examples are repeated at different scales, but for clarity these are not drawn here; see text for details.

Instead, we use a sparse pattern of training examples as illustrated in Figure 2.9. For an object in the training set at location $(\mathbf{x}, s)$, positive examples are taken at the 3x3x3 scaled grid locations $\mathbf{x}' = \mathbf{x} + [z_{\mathrm{x}} s' \delta_1, z_{\mathrm{y}} s' \delta_1]^T$ for scales $s' = s\gamma_1^{z_{\mathrm{s}}}$, where $(z_{\mathrm{x}}, z_{\mathrm{y}}, z_{\mathrm{s}}) \in \{-1, 0, +1\}^3$. The grid is spaced by δ_1 in (scale-normalized) space and γ_1 in scale. One scale, $z_{\mathrm{s}} = 0$, of this grid is shown in Figure 2.9(a). The positive examples ensure a strong classification response near the true centroid, wide enough that the sliding window need not be evaluated at every pixel (see Section 2.3.2). To ensure the response is localized, negative examples (the outer grid of red circles in Figure 2.9(a)) are taken at positions $\mathbf{x}' = \mathbf{x} + [z_{\mathrm{x}} s' \delta_2, z_{\mathrm{y}} s' \delta_2]^T$ for scales $s' = s\gamma_2^{z_{\mathrm{s}}}$, with a larger spacing $\delta_2 > \delta_1$ and scaling $\gamma_2 > \gamma_1$, and using the same $(z_{\mathrm{x}}, z_{\mathrm{y}}, z_{\mathrm{s}})$ though now excluding $(0, 0, 0)$. This particular pattern results in a total of 53 examples for each object; while this may seem a large number it is vastly fewer than the total number of scale-space locations in the image. For training images not containing an object, we create (all negative) examples in the same pattern, at a number of random scale-space locations, as illustrated in Figure 2.9(b).

For the size of learning problems addressed in this chapter, feature vectors can be precomputed for all examples. This usually takes less than an hour depending on the dataset. The boosting procedure is then relatively quick, taking typically less than a minute to con-

verge, since the weak learners are individually quite powerful.

2.4.2 Classification Cascade

The ordering of the weak learners selected by boosting is important: early rounds select more general weak learners which classify the bulk of the training examples well, while later rounds concentrate on more particular troublesome examples. This ordering can be exploited for efficiency at test time by building a cascade [Viola & Jones, 2001].

We use a very simple form of cascade, similar to that in [Schneiderman & Kanade, 2004]: after each round m of boosting, a threshold ρ_m is chosen as the minimum classification confidence value H_m across all positive training examples:

$$\rho_m = \min_{i \text{ s.t. } z_i=+1} H_m(\mathbf{x}_i, s_i) - \varepsilon \ , \tag{2.21}$$

with a small constant ε subtracted to aid generalization. Threshold ρ_m is used at test time, for each round, to determine locations which are very unlikely to be true detections: if, at round m, the classification confidence $H_m(\mathbf{x}, s)$ is less than ρ_m, then location $(\mathbf{x}, s)$ is removed from further consideration so that weak learners for rounds $m' > m$ are not evaluated there. Note that ρ_m is the largest threshold to give no false negatives across the training set. The cascade gives a considerable speed-up, with almost no performance degradation. On real data we have observed that the average number of rounds computed per point drops to about 20%, giving about a 2x speed-up in detection time per image.[1]

2.4.3 Retraining on Training Data

It is unclear how to place the sparse negative training examples optimally throughout the training images, and hence initially they are placed at random, as described above. However, once a detector is learned using these examples, a retraining step is used to boot-strap the set of training examples, in a similar manner to [Zhu & Ghahramani, 2002]. We evaluate the detector on the training images, and record all detections not marked as correct (as defined in Section 2.5.1) and any false negatives. The classifier is then retrained on the original example set, augmented with new negative examples at the locations of incorrect detections, and duplicate positive examples to correct the false negatives. As we demonstrate in

[1] For implementation reasons, the speed-up is not directly inversely proportional to the drop in the average number of rounds computed.

Section 2.5.4, this procedure allows us to learn the parameters of more parts without overfitting. In [Shotton *et al.*, 2005] we referred to a very similar procedure as *partially supervised learning*.

2.4.4 Retraining on Test Data

The same idea can be put to work on the *test* data, if one assigns a degree of trust to the output of the classifier. One can take a fixed proportion ξ (e.g. $\xi = 10\%$) of detections with strongest confidence and assume these are correct, positive detections, and the same proportion of detections with weakest confidence and assume there are no objects present at those locations. The boosted classifier is learned again with the new positive and negative training examples further augmenting the training set.

2.5 Evaluation

In this section we present a thorough evaluation of our technique on several challenging datasets. Our technique is applied to the problems of classification and detection. We investigate the performance of different aspects of our system, and compare against other state-of-the-art methods. The standard experimental procedure is detailed in Section 2.5.1, the datasets in Section 2.5.2, and the results begin in Section 2.5.3.

2.5.1 Procedure

In each experiment, the image datasets are split into training and test sets. Each model is learned from the training set with ground truth bounding boxes provided. At test time, the bounding boxes are only used to compute detection accuracy, as follows.

The mode detection procedure, described in Section 2.3, results in a set of centroid hypotheses and confidence values for object presence at these points. We assign a scaled bounding box centered on each detection, with aspect-ratio proportional to that of the average training bounding box. For a detection to be marked as correct, its inferred bounding box b_{inf} must agree with the ground truth bounding box b_{gt} based on an overlap criterion (as used in [VOC]):

$$\frac{\text{area}(b_{\text{inf}} \cap b_{\text{gt}})}{\text{area}(b_{\text{inf}} \cup b_{\text{gt}})} > 0.5 \,. \tag{2.22}$$

Each ground truth bounding box can match against only one inferred bounding box, so that spurious detections of the same object count as false positives. For the task of image

classification, we take the single most confident detection within each image, and use its confidence as the classification confidence. The nature of the mean shift detection procedure ensures that every image has at least one detection.

For the task of classification, we use the receiver operating characteristic (ROC) curve to measure performance. This plots the trade-off between false positives and false negatives as a global confidence threshold is applied. The equal-error rate (EER) gives an easy-to-interpret measure of quality of classification, while the area under the curve (AUC) which takes the whole curve into account gives a better measure for comparison purposes.

For detection we use two closely related measures. The first, the recall-precision (RP) curve, plots the trade-off between recall and precision as one varies the global threshold. Where necessary for comparison with previous work, we use the EER measure on the RP curve, though wherever possible we use the more representative AUC measure.[2] The second measure plots recall against the average number of false positives per image (RFPPI) as the detection threshold is varied [Ferrari *et al.*, 2006a]. The RFPPI curve seems more natural for human interpretation than the RP curve, since it is monotonic and stabilizes as more negative images are tested (the RP curve can only deteriorate). Note that the legends in Figures 2.10, 2.13(b), 4.4(b), and 4.7(b) contain RP AUC figures even though the graphs show RFPPI.

2.5.2 Datasets

We specify here the datasets used in the evaluations below. Bounding boxes are used, but apart from Section 2.5.7, no figure-ground segmentations are used (a few were used in [Shotton *et al.*, 2005]). Example images from the datasets are shown in Appendix C.

Weizmann Horses

The Weizmann horse database [Weizmann] is a very challenging set of side-on horse images. Used for evaluating segmentation accuracy in [Borenstein *et al.*, 2004], we introduced it for evaluation of detection in [Shotton *et al.*, 2005]. A wide variety of horses breeds, colors, and textures are represented, with different articulations, lighting conditions and scales. While they are nominally viewed side-on and facing left, considerable out-of-plane rotation is evident.[3]

[2] Strictly speaking, it is the area to the *right* of the curve in a recall against $1-$precision plot.

[3] Although not required here, it is fairly simple to extend our algorithm to allow the detection of objects facing in both directions. The standard detector is evaluated on both the original image and a left-right mirrored copy

In [Shotton *et al.*, 2005], we evaluated our detector against a single-scale version of the database paired with background images from the Caltech database [Caltech]: 50 horse images (10 of which were segmented) and 50 background images were used for training, and the remaining 277 horse images and 277 background images for testing.

In this work, we extend the evaluation to a multi-scale version of this image database, and to improve the quality of the benchmark, use a much harder set of background images from the Caltech 101 dataset [Caltech 101] and [Fei-Fei *et al.*, 2006]. While these image sets have very different textural characteristics (and hence we would expect texture-based methods to work well at classification), the background images containing lots of clutter edges pose a hard challenge to our contour-only detector. All images were down-sampled to a maximum image dimension of 320 pixels where necessary; the resulting horses have a scale range of roughly 2.5 from smallest to largest. For this dataset, the first 50 images from horse and background sets were used for training, the next 50 as a validation set for optimizing parameters, and a final 228 as the test set. We have made both the multi-scale and single-scale datasets available from our website at [Shotton].

Graz 17

We compare our method against the results obtained by [Opelt *et al.*, 2006c] on their 17 class database (listed in Table 2.1). We use the same training and test sets, including for training the 'validation' set, which is integral to the learning algorithm of [Opelt *et al.*, 2006c]. Images are down-sampled to a maximum image dimension of 320 pixels where necessary. For some classes, the resulting scale range is more than 5 times from smallest to largest. We investigate each class individually and evaluate against the class test set and an equal number of images from the background test set (where possible, since only 166 background images are in the dataset). We make the comparison as fair as possible, though for some classes the number of training and test images quoted in the paper vary slightly to those in the online dataset.

2.5.3 Matching Measures

We now turn to the results, beginning by comparing the performance of the detector using several different matching measures: our proposed oriented chamfer matching with learned

of the image. The detections from both images are then combined while removing duplicates. See e.g. [Shotton *et al.*, 2005].

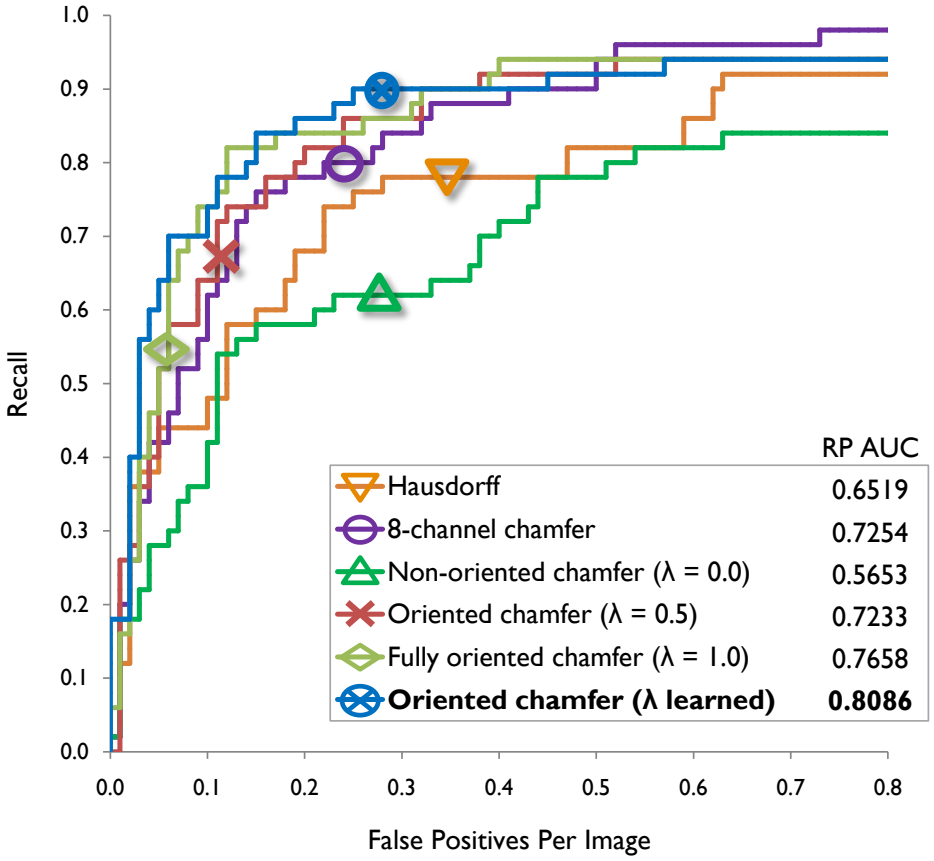

Figure 2.10: **Detection performance of different contour matching measures.** Recall is plotted as a function of the number of false positives per image averaged over the dataset. We observe the best performance is obtained by our oriented chamfer matching technique with a learned λ parameter, although a fixed $\lambda = 1$ also performs well.

λ and with constant $\lambda \in \{0, 0.5, 1\}$, standard 8-channel chamfer matching, and Hausdorff matching. The experiment was performed against 100 images in the multi-scale Weizmann test set using 100 parts without retraining.

Figure 2.10 superimposes the RFPPI curves for each matching measure, and the legend reports the corresponding RP AUC statistics. Observe that with no orientation information ($\lambda = 0$, identical to a 1-channel, non-oriented chamfer distance), performance is very poor. Hausdorff distance also fails to work well, since it too does not use orientation information. The 8-channel chamfer matching performs fairly well, but by modeling the orientation explicitly, our oriented chamfer distance (for $\lambda > 0$) performs as well or better, even if λ is kept constant. The RFPPI curve for $\lambda = 1$ appears almost as good as the learned λ curve, although the AUC numbers confirm that learning λ per part, in order to weight optimally between distance and orientation in (2.8), is noticeably better.

2.5.4 Retraining

As described in Sections 2.4.3 and 2.4.4, one can boot-strap the detector by retraining firstly on the training data to prevent the detector firing in background clutter, and secondly on the test data by assigning a degree of confidence ξ to the results. For this experiment we recorded the performance of the detector (RP AUC) against the number of parts: (i) without retraining, (ii) retraining only on the training data (identified in Figures 2.11 and 2.13 as 'retrained training'), and (iii) retraining both on the training and test data (identified as 'retrained test'), against the multi-scale Weizmann validation dataset. The confidence parameter was set to $\xi = 10\%$.

We can draw several conclusions from the plot of these results in Figure 2.11 (the slight noise is due to the considerable impact that even one false negative has on the RP AUC). Adding more parts (by performing more rounds of boosting) helps performance on the test data up to a point, but eventually the detector starts to over-fit to the training data and generalization decreases. By providing more training examples, by retraining on the training data, we can use more weak learners without over-fitting (though of course over-fitting will recur eventually), and obtain improved detection performance at the expense of more parts. Retraining on both the training and test data allows a further improvement. Note that with fewer parts (40), retraining in either way can actually decrease performance, since the strongest and weakest detections are not sufficiently reliable. Note also the significant extra effort that retraining entails, for the relatively small performance gain.

2.5.5 Approximate Chamfer Matching

The results of our evaluation make use of the approximation described at the end of Section 2.2.1, whereby only a subset of fragment edgels are used for chamfer matching. For all experiments, only every fifth edgel (sorted in scan-line order) in each fragment is used, giving a commensurate speed improvement. To determine whether this approximation adversely affects performance, we compare detection performance with and without the approximation, on the Weizmann multi-scale validation dataset using 100 features. With the approximation, a RP AUC of 0.9547 was achieved, whereas without the approximation (matching every edgel) only 0.9417 was obtained. We conclude that the approximation can improve speed without degrading detection performance. The slight improvement in performance may even be significant, since the variance of the part responses in the training

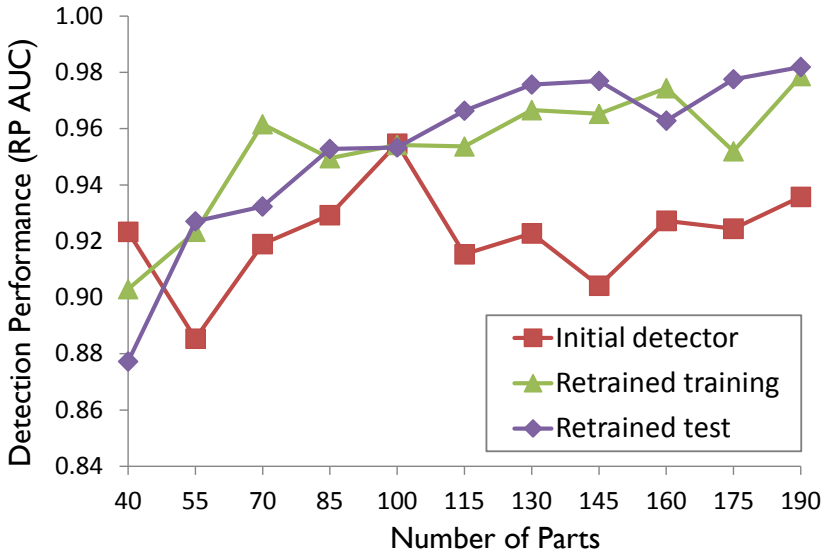

Figure 2.11: **Performance before and after retraining.** Detection performance is graphed as a function of the number of parts (rounds of boosting) for the initial detector and for the detectors when retrained either on the training dataset or on the training and test datasets. While the initial detector starts to over-fit as the number of parts is increased above 100, retraining prevents over-fitting (over the range of the graph), allowing an overall performance improvement at the expense of more parts.

data is increased slightly, which may prevent over-fitting.

2.5.6 Multi-Scale Weizmann Horses

We now evaluate on the full Weizmann multi-scale dataset, showing example detections in Figure 2.12 and quantitative results for classification and detection in Figure 2.13. With retraining, we achieve a final ROC AUC of 0.9400 for classification, and a final RP AUC of 0.8903 for detection.

There are several conclusions to draw from these results. Firstly, we have confirmed the results of Section 2.5.4 that retraining on the training and test sets can improve performance. Next, turning to the correct and incorrect detections in Figure 2.12, we observe that the detector works very well on the extremely challenging horse images, despite wide within-class variation, extensive background clutter and some extreme lighting conditions. Missed detections (false negatives) have occurred when there is significant pose change or out-of-plane rotation beyond the range for which we would expect our side-on detector to work. Training explicitly for these poses or rotations, perhaps using a multi-class classifier such as [Torralba *et al.*, 2007], should allow detection of these objects. False positives occur when the pattern of clutter edgels is sufficiently similar to our model, as for example the case

Figure 2.12: **Example detections in the multi-scale Weizmann horse test set.** Bounding boxes around objects indicate detections: green represents correct detections, red false positives, and yellow the ground truth for false negatives. The final column visualizes the contour fragments for the detections of the penultimate column. Note accurate scale-space localization in the presence of highly variable object appearance, significant background clutter, extreme lighting conditions (including silhouetting), articulation, and pose changes.

(middle column, penultimate row) of the man standing in front of the horse, where the man's legs look sufficiently similar to the front legs of a horse in terms of image edges. We show in Chapter 4 that simple texture based features are sufficient to discount many such false positives, and that the combination of contour and texture based features significantly improves performance. For object classes such as horses with very distinctive contour but variable texture, we show that contour gives important cues about where the object *is*, while texture gives strong cues about where the object is *not* (for example, blue sky or green grass is unlikely to signify horse presence).

Our C# implementation on a 2.2 GHz machine takes approximately 2 hours to train

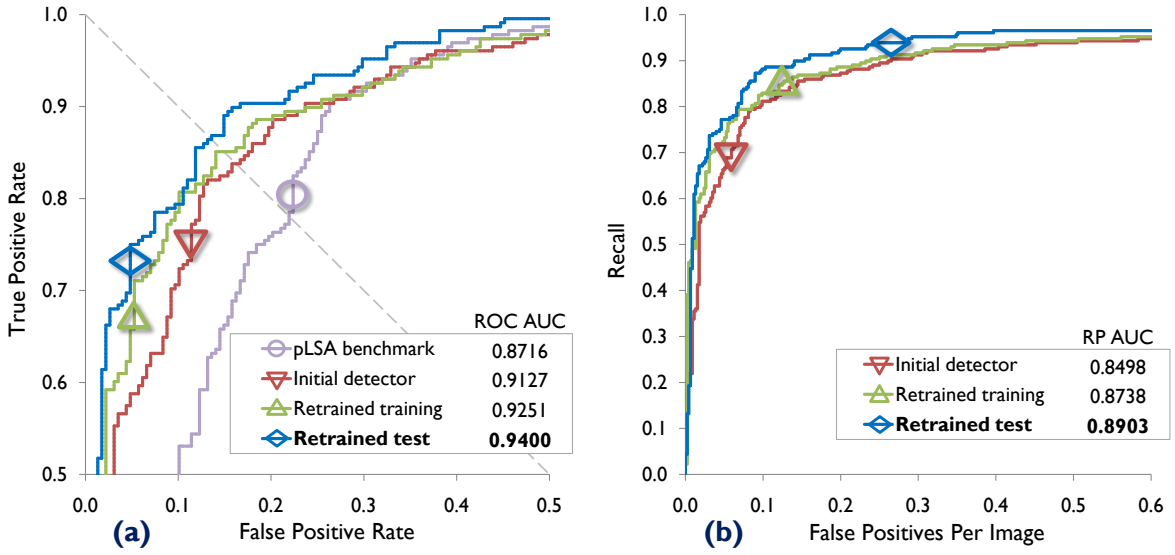

Figure 2.13: **Performance on the Weizmann horse test set with and without retraining. (a)** ROC curves showing classification performance, with the curve for the pLSA classification benchmark included (Section 2.5.9). To aid readability, only the top-left corner is shown. **(b)** RFPPI curves showing detection performance. Note how both stages of retraining improve both classification and detection performance.

and 10 seconds per image to test. For these and all other experiments, unless stated otherwise, the following parameters were used. The distance transform truncation was $\tau = 30$, and fragments were randomly chosen with the following transformation parameters: scaling $s_{\text{rnd}} = 1.2$, rotation about fragment center $\theta_{\text{rnd}} = \frac{\pi}{8}$, (scale-normalized) translation $t_{\text{rnd}} = 0.05$, and rotation about centroid $\phi_{\text{rnd}} = \frac{\pi}{16}$. To learn the dictionary, 10000 raw fragments, with edgel density bounded as $(\eta_1, \eta_2) = (1\%, 5\%)$, were clustered using a constant $\lambda = 0.4$, to produce 500 exemplars. To learn the classifier, examples were taken with grid spacings $\delta_1 = 0.03$ and $\delta_2 = 0.25$, and grid scale scalings $\gamma_1 = 1.1$ and $\gamma_2 = 1.4$. Three patterns of negative examples were used for background images, and λ was allowed values in $\{0, 0.2, \ldots, 1\}$. Evaluation took place with the cascade constant $\varepsilon = 3$, and used a grid spacing of $\Delta_{\text{grid}} = 0.07$ scaled by each of 6 test scales over $M = 100$ rounds. The top and bottom $\xi = 10\%$ of detections were used for retraining on the test set.

2.5.7 Training from Segmented Data

To investigate whether the contour codebook learned from unsegmented data works well, we performed the same detection experiment on the multi-scale Weizmann dataset but now with the codebook learned from segmented training data. We obtained a detection RP AUC of 0.8637, slightly worse than the performance on unsegmented images (0.8903). This some-

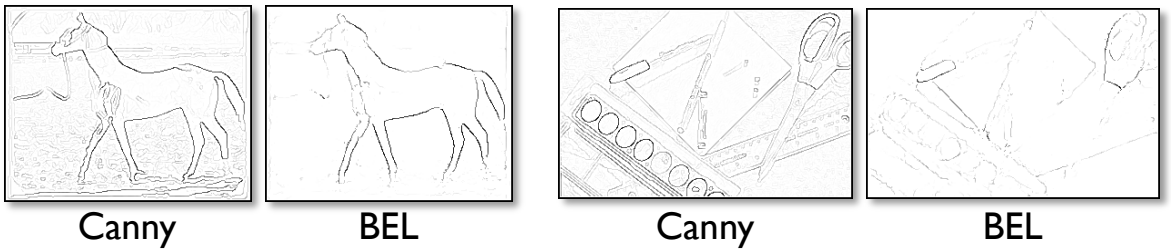

Figure 2.14: **Example edge maps from Canny [Canny, 1986] and boosted edge learning (BEL) [Dollár *et al.*, 2006].** One horse image, and one background image are shown. By learning a model of horse edges, BEL is able to produce edge maps with visibly less clutter. We demonstrate how this substantially improves detection performance in Section 2.5.8.

what unexpected result, possibly due to over-fitting or to the importance of interior edgels, shows the strength of our algorithm for learning the codebook.

2.5.8 Learned Edge Detection

The Canny edge detector [Canny, 1986] has thus far proved a capable basis for our features. However, recent developments such as [Martin *et al.*, 2004; Dollár *et al.*, 2006] take a more modern approach to edge detection, whereby a model of edges is learned from training data. To determine whether the choice of edge detector has a significant impact, we compared performance on the multi-scale Weizmann dataset using two models trained by the boosted edge learning (BEL) algorithm from [Dollár *et al.*, 2006]. The first was trained on a set of natural images with corresponding hand-drawn edge labels, and the second on our segmented horse training set with the aim of detecting only horse edges. The edge maps returned by [Dollár *et al.*, 2006] are soft, and so standard non-maximal suppression is used to give 'thin' edges. The image edge gradient is used to obtain edge orientation information, just as for Canny edges.

The detection results were as follows. For 100 parts, without retraining, the Canny edge detector gives us a RP AUC of 0.8498. The first BEL model, trained on natural images, gave no improvement with a RP AUC of 0.8354. However, the second BEL model, trained on segmented horse images, gave 0.8976 RP AUC, a very significant improvement, and even slightly better than the best performance using Canny with retraining (0.8903). There is an even more noticeable improvement in classification performance: from 0.9127 ROC AUC for Canny, up to 0.9518 for the BEL. We attribute this improvement to the reduced number of clutter edges found by the BEL: even to the naked eye, there is a marked difference in edge density between the horse and background images (illustrated in Figure 2.14).

Of note in the qualitative results was that the learned codebook contained noticeably cleaner contour fragments than those learned from Canny edge maps, with very little noise. Most of the quantitative improvement was due to several detections that had previously been missed. This experiment has confirmed that a modern learned edge detection algorithm complements our object detection system; future work remains to extend this evaluation to the other datasets in this chapter.

2.5.9 Comparison with Sparse Local Descriptors

To compare contour fragments with sparse local descriptors, and to determine the challenge that the multi-scale Weizmann horse dataset poses to them, we evaluated a benchmark using probabilistic latent semantic analysis (pLSA) [Sivic *et al.*, 2005; Hofmann, 2001], adapted to give image classification. With such a wide variety of texture- and interest point-based methods in the literature it would be impossible to evaluate them all; bag-of-words models were shown to perform best in the PASCAL Visual Object Challenge 2006 [VOC], and we choose pLSA as a simple-to-implement but powerful modern representative of these, to give us an indicative comparison.

The technique takes SIFT descriptors [Lowe, 2004] and clusters them into a number of visual *words* w so that each image (or *document*) d is represented by the counts of words present. The unsupervised pLSA algorithm then mines the word-document co-occurrence table looking for recurring *topics* k.

We first run pLSA on both training and test data combined, resulting in a distribution $p(k|d)$ for each image. For the training images, we know $p(\text{obj}|d) \in \{0, 1\}$ where the event 'obj' denotes the presence of an object in the image, and so can learn distributions $p(\text{obj}|k) \propto \sum_d p(k|d)p(\text{obj}|d)p(d)$ (using uniform priors). Hence for test images we can compute the posterior classification as $p(\text{obj}|d) = \sum_k p(\text{obj}|k)p(k|d)$.

To obtain the best performance this model would allow, we optimized the parameters against the test set, giving 200 SIFT clusters and 15 topics shared between the object and background classes. In Figure 2.13(a) we plot the ROC curve for the pLSA benchmark. We observe considerably worse performance than our contour based classifiers achieve, suggesting that these images are difficult to classify based on sparse local descriptors alone.

2.5.10 Single-Scale Weizmann Horses

Using the original single-scale Weizmann horse dataset, we compare our results with those of [Shotton *et al.*, 2005], where a RP EER of 92.1% was achieved (using some segmented data). Further experiments on this dataset in [Ferrari *et al.*, 2006a] improved on this figure with an RP EER of 94.2% for contour based features only, and 95.7% combining contour features and local descriptors. Our improved method presented in this chapter, using only contour and without segmented training data, obtains an RP EER of 95.68% (with a corresponding RP AUC of 0.9496), as good as [Ferrari *et al.*, 2006a] even though they employ an additional feature type. We speculate that the improved performance over our previous method is due to the better generalization given by our learned codebook, and the use of the mean shift algorithm to select detections.

2.5.11 Graz 17

	Number of images		Classification (ROC)		Detection (RP)		
Class	Training	Test	AUC	EER	AUC	**EER**	Opelt EER
Airplanes	100	400	0.9953	3.4%	0.9310	**6.8%**	7.4%
Cars (rear)	100	400	0.9992	1.5%	0.9912	**1.8%**	2.3%
Motorbikes	100	400	1.0000	0.4%	1.0000	**0.3%**	4.4%
Faces	100	217	0.9966	2.4%	0.9850	**2.8%**	3.6%
Bikes (side)	90	53	0.9366	13.2%	0.6959	**32.1%**	28.0%
Bikes (rear)	29	13	0.9172	15.4%	0.6398	**26.7%**	25.0%
Bikes (front)	19	12	0.9375	16.7%	0.6344	**41.7%**	41.7%
Cars ($\frac{2}{3}$ rear)	32	14	0.9000	20.9%	0.6925	**30.0%**	12.5%
Cars (front)	34	16	0.9727	12.5%	0.7233	**29.4%**	10.0%
Bottles	54	64	0.9802	7.8%	0.9468	**9.4%**	9.0%
Cows (side)	45	65	0.9992	1.7%	0.9975	**1.5%**	0.0%
Horses (side)	55	96	0.9816	6.3%	0.9680	**6.3%**	8.2%
Horses (front)	44	22	0.9566	13.6%	0.7852	**27.3%**	13.8%
Cows (front)	34	16	0.9727	6.3%	0.8575	**18.8%**	18.0%
People	39	18	0.9321	16.7%	0.4271	**47.6%**	47.4%
Mugs	30	20	0.9600	5.0%	0.9035	**10.0%**	6.7%
Cups	31	20	0.9825	5.0%	0.9158	**15.0%**	18.8%

Table 2.1: **Classification and detection performance on the Graz 17 dataset.** The final column compares detection performance with [Opelt *et al.*, 2006c].

We conclude our evaluation by investigating performance on the Graz 17 class dataset. Our results are compared to [Opelt *et al.*, 2006c] in Table 2.1, and Figures 2.15 and 2.16 show example detections. Parameter values were unchanged from the previous multi-scale Weizmann experiments, although the number of parts and number of scales were adjusted against the training data.

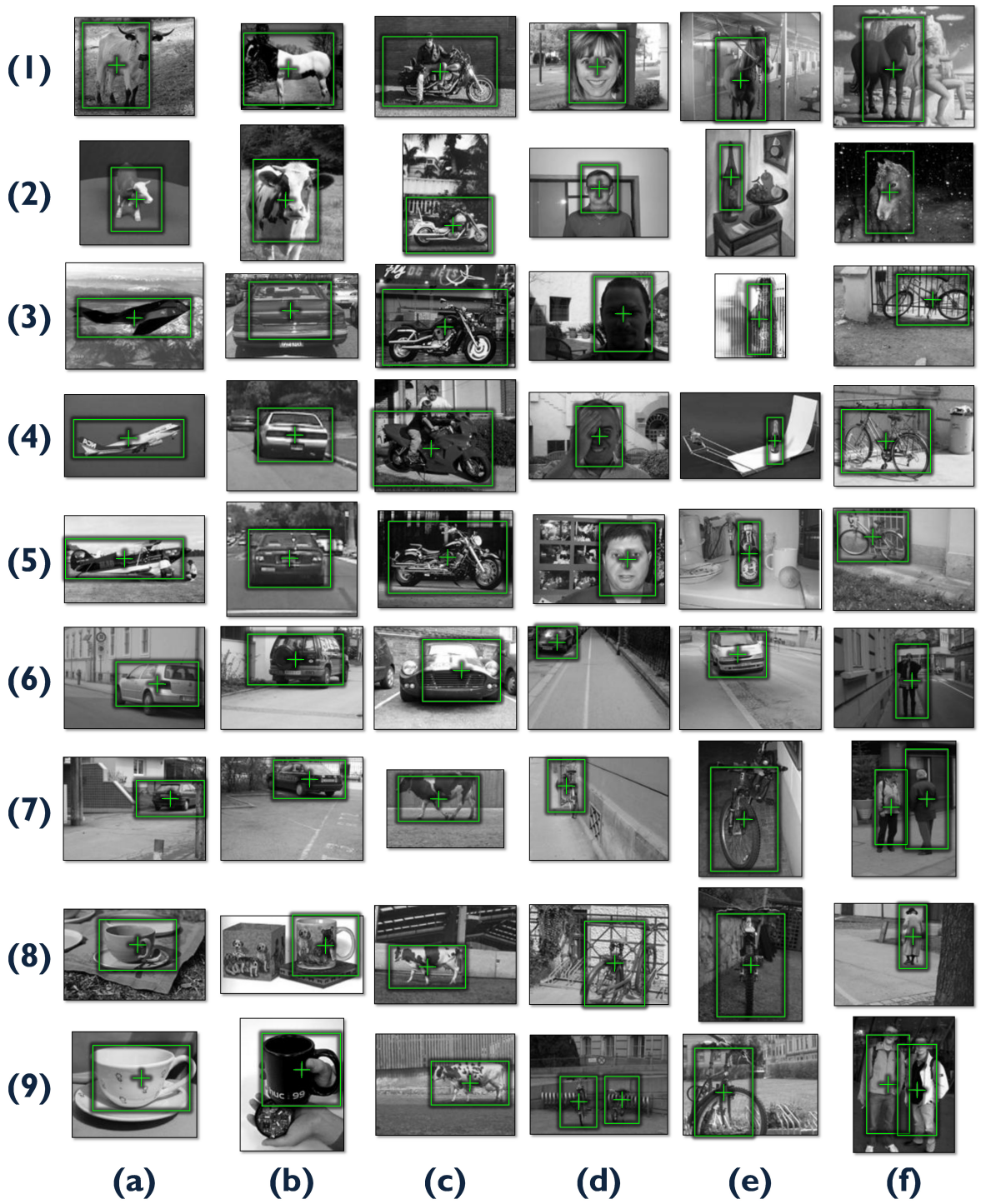

Figure 2.15: **Example detections for the Graz 17 class test set.** Green bounding boxes around objects indicate detections. Note accurate scale-space localization of objects despite wide within-class appearance variation, significant pose changes, partial occlusion, and background clutter, and detection of multiple objects.

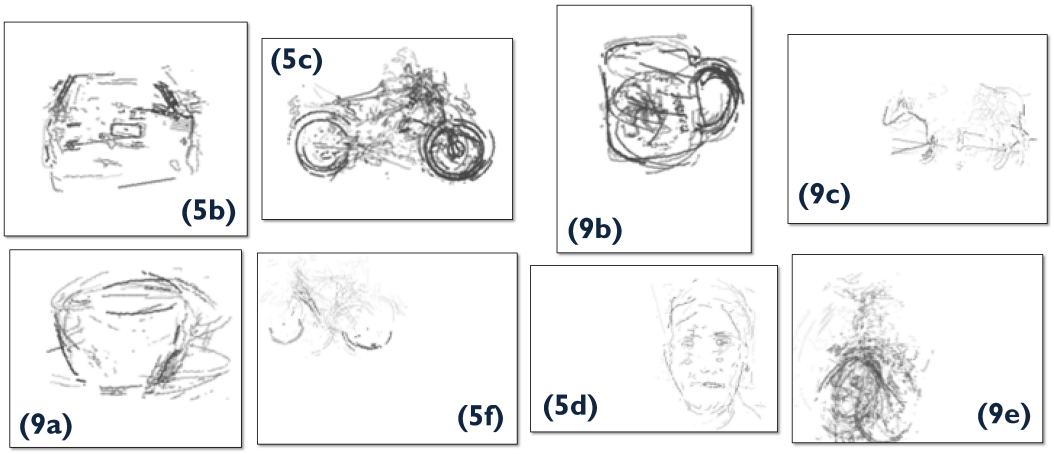

Figure 2.16: **Example contour visualizations from detections in Figure 2.15.** These visualizations superimpose, at their detected positions, all contour fragments that contributed positively to the object detections. This gives an indication of which parts of object contour are useful for recognition. For example, in (9c), the cow is recognized using head and rear contour fragments with little useful information used from its back.

There are several conclusions to draw from these results. Firstly, in almost every case we perform comparably to [Opelt *et al.*, 2006c], and for the larger datasets we show a significant improvement, with almost perfect performance on motorbikes. As one would expect, classification proves easier than detection in most cases, since strong but poorly localized detections contribute positively to classification but negatively to detection. Performance is worse for a few classes, such as cars ($\frac{2}{3}$ rear) and cars (front), and poor for both techniques for bikes (front) and people. There are few training images of these classes, and objects exhibit considerably more out-of-plane rotation. Also, the small number of test images means that even one missed detection has a very large effect on the RP EER (up to $\frac{100}{N}\%$ for N test images). Much more significant therefore is our sustained improvement for classes with more test images.

2.6 Conclusions

Our thorough evaluation has demonstrated that contour can be used to successfully recognize objects from a wide variety of object classes at multiple scales. Our new approximate oriented chamfer distance outperformed existing contour matching methods, and enabled us to build a class-specific codebook of local contour fragments, even without segmented training data. We observed that retraining on both the training and test data can improve generalization and test performance. Finally, we showed how modern, learned edge detection gave an improvement over the traditional Canny edge detector.

CHAPTER 3

TEXTURE

3.1 Introduction

In this chapter, we turn our attention to the problem of automatically achieving semantic segmentations of photographs. We propose a system, called *TextonBoost*, that automatically partitions a given image into semantically meaningful regions, each labeled with a specific object class, as illustrated in Figure 3.1.

The challenge is to model the visual variability of a large number of both structured and unstructured object classes, to be invariant to viewpoint and illumination, and to be robust to occlusion. Our focus is not only the accuracy of segmentation and recognition, but also the efficiency of the algorithm, which becomes particularly important when dealing with large image collections or video sequences.

At a local level, the *appearance* of an image patch leads to ambiguities in its class label. For example, a window could be part of a car, a building or an airplane. To overcome these ambiguities, it is necessary to incorporate longer range information such as the spatial configuration of an object (the object *shape*) and also *contextual* information from the surrounding image. To achieve this, we construct a discriminative model for labeling images which exploits all three types of information: appearance, shape, and context. Our technique can model very long-range contextual relationships extending over half the size of the image.

Additionally, our technique overcomes several problems typically associated with object recognition techniques that rely on sparse features (such as [Lowe, 2004; Mikolajczyk & Schmid, 2002]). These problems are mainly related to textureless or very highly textured image regions. Figure 3.2 shows some examples of images with which those techniques would very likely struggle. In contrast, our technique based on dense features is capable of coping with both textured and untextured objects, and with multiple objects which inter- or

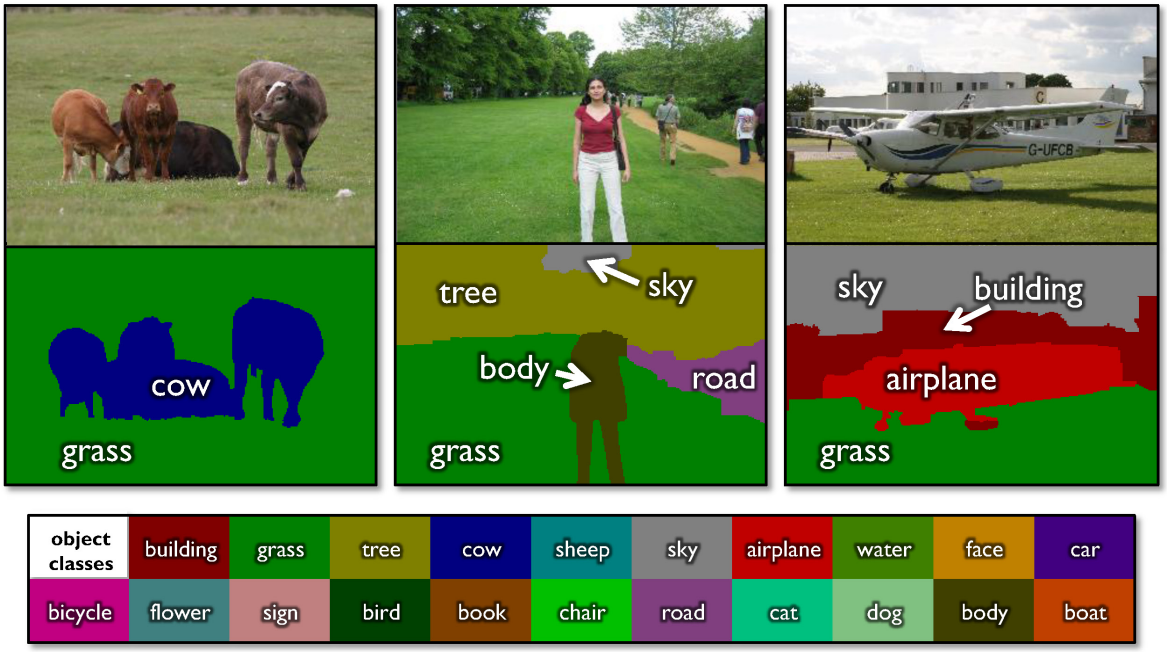

Figure 3.1: **Example results of our new simultaneous object class recognition and segmentation algorithm.** Up to 21 object classes (color-coded in the key) are recognized, and the corresponding object instances segmented in the images. For clarity, textual labels have been superimposed on the resulting segmentations. Note, for instance, how the airplane has been correctly recognized and separated from the building, the sky, and the grass lawn. In these experiments only one learned multi-class model has been used to segment all the test images. Further results from this system are given in Figure 3.21.

self-occlude, while retaining high efficiency.

The contributions in this chapter are threefold. First, we present a discriminative model which is capable of fusing shape, appearance and context information to recognize efficiently the object classes present in an image, whilst exploiting edge information to provide an accurate segmentation. Second, we propose features, based on textons, which are capable of modeling object shape, appearance and context. Finally, we demonstrate how to train the model efficiently on a very large dataset by exploiting both boosting and piecewise training methods.

The TextonBoost system originally appeared in [Shotton *et al.*, 2006]. This chapter builds on that work, with clarified explanations, new variants of our texture-based features, and an extended evaluation.

The chapter is organized as follows. We briefly discuss closely related work below. In the Section 3.2, we describe the image databases used in our experiments. Section 3.3 introduces the high-level model, a conditional random field (CRF), while Section 3.4 presents our novel low-level image features and their use in constructing a boosted classifier. Experiments,

Figure 3.2: **Example problem images for techniques based on sparse features.** Sparse feature based techniques struggle with textureless and very highly textured regions, and multiple objects, especially those that severely inter-occlude.

performance evaluations and conclusions are given in the final two sections. Note that the notation in this chapter should be treated separately from that of Chapter 2. An overview of the TextonBoost system is shown in Figure 3.3; the terms in this figure will become clear as the reader continues through the chapter.

Related Work

The reader is referred to Appendix A for a fuller discussion of related work, but we briefly highlight some directly related research here.

[Duygulu *et al.*, 2002] used a classifier, trained from images with associated textual class labels, to label regions found by bottom-up segmentation. Such segmentations often do not correlate with semantic objects, for example an object in shadow may be divided into a shadowed versus non-shadowed part. Our solution to this problem is to perform segmentation and recognition in the same unified framework rather than in two separate steps. Such a unified approach was presented in [Tu *et al.*, 2003], but there only text and faces were recognized, and at a high computational cost. In [Konishi & Yuille, 2000], images were labeled using only a unary classifier and hence did not achieve spatially coherent segmentations.

The most similar work to ours [He *et al.*, 2004] incorporates region and global label features to model shape and context in a conditional random field. Their work uses Gibbs sampling for both the parameter learning and label inference, and is therefore limited in the size of dataset and number of classes that can be handled efficiently. Our focus on the speed of training and inference allows us to use larger datasets with many more object classes. We currently handle 21 classes (compared to the seven classes of [He *et al.*, 2004]); it would be tractable to train our model on even larger datasets than presented here.

More recent work [He *et al.*, 2006] by the same group presented a related technique, where images are first segmented with a bottom-up algorithm to give 'super-pixels' which

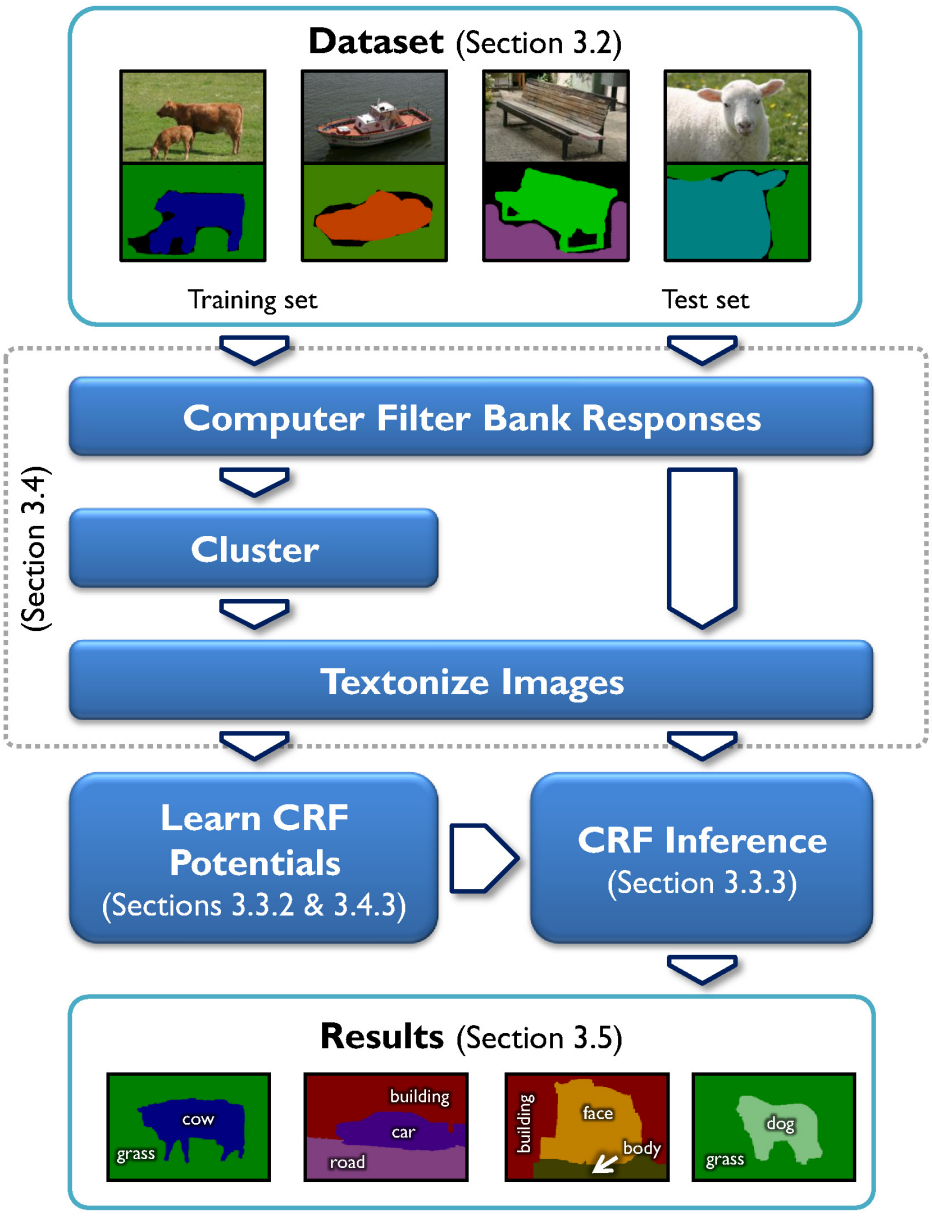

Figure 3.3: **Overview of the TextonBoost algorithm presented in this chapter.** A set of images with associated ground truth segmentations is divided into training and test sets. The responses of a filter bank convolved with the training images are clustered, and the resulting clusters are used to textonize the images (see Section 3.4.1). The parameters of the potentials in the CRF are then learned. Finally, inference is run on the test images, giving semantic segmentations as outputs.

are then merged together and semantically labeled using a combination of several scene-specific CRF models. Their technique improved slightly the quantitative results from [He *et al.*, 2004], but still has not been demonstrated to handle more than 11 classes. Additionally, the super-pixelization is a hard decision from which their CRF-based model cannot recover. Our technique instead works efficiently at a per-pixel level.

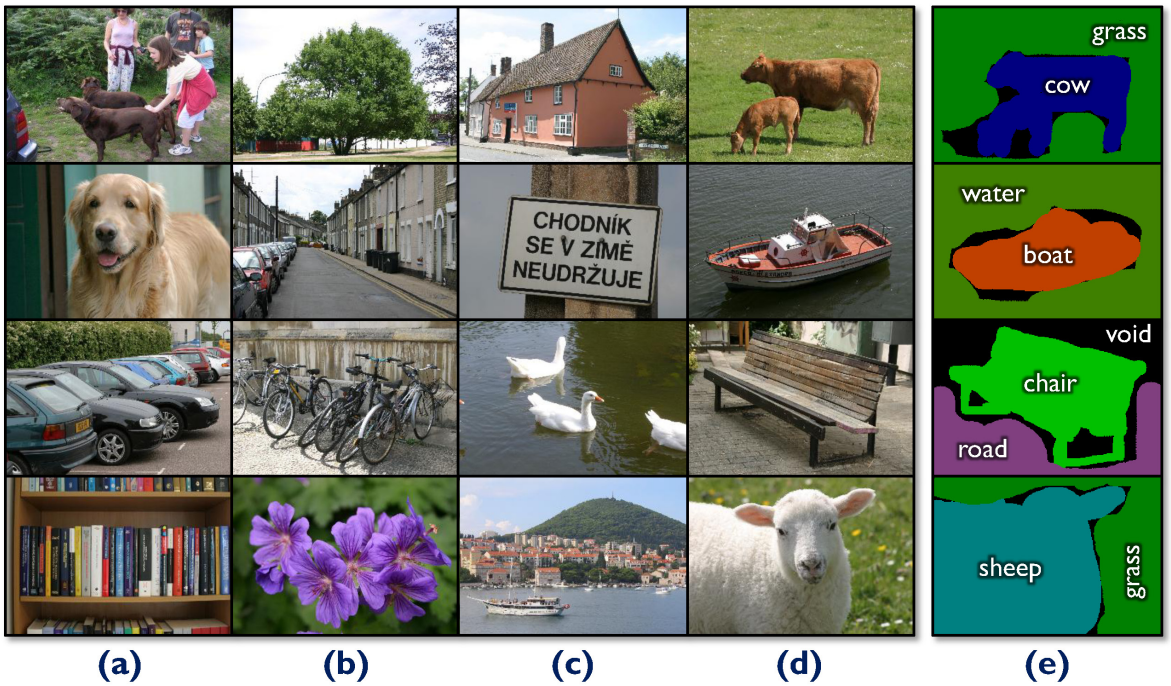

Figure 3.4: **The MSRC labeled image database. (a-d)** A selection of images in the 21-class database. **(e)** The ground truth annotations corresponding to column (d). Each color maps uniquely to an object class label. All images are approximately 320×240 pixels.

3.2 Image Databases

Our object class models are learned from a set of labeled training images. In this chapter we consider four different labeled image databases. The Microsoft Research Cambridge (MSRC) database, available at [MSRC 21], is composed of 591 photographs of the following 21 object classes: building, grass, tree, cow, sheep, sky, airplane, water, face, car, bicycle, flower, sign, bird, book, chair, road, cat, dog, body, boat. Examples are shown in Figure 3.4. The training images were hand-labeled by means of a 'paint' interface, with the assigned colors acting as indices into the list of object classes. One could instead use one of the novel 'user-centric computation' methods such as [Russel *et al.*, 2005] or [Peekaboom]. Note that we consider general lighting conditions, camera viewpoint, scene geometry, object pose and articulation. Our database is split randomly into roughly 45% training, 10% validation and 45% test sets. This is done per class to ensure approximately proportional contributions from each class.

Note that the ground truth labeling of the 21-class database contains pixels labeled as void. These were included both to cope with pixels that do not belong to a class in the database, and also to allow for a rough and quick hand-segmentation which does not align exactly with the object boundaries. Due to this ambiguity in meaning, it was not sensible to

learn a background class based on these regions, and hence void pixels are ignored for both training and testing.

For comparison with previous work [He *et al.*, 2004], we also used the 7-class Corel database subset (where images are 180×120 pixels) and the 7-class Sowerby database (96×64 pixels). For those two databases, the numbers of images in the training and test sets we used are exactly as for [He *et al.*, 2004], although their precise train-test split was not known. Neither of these data sets include the void label.

The final evaluation we present was performed on a set of nine 20-minute video sequences of Japanese television programs: modern drama, news, golf, soccer, cooking, variety, music, historical drama, and business news. The set of classes used for this evaluation was as for the MSRC evaluation, but without sign, book or chair, and including the new classes hand, table and headgear. For speed of evaluation, video frames were downsampled to 336x224 pixel resolution, but were otherwise not preprocessed. For both training purposes and quantitative evaluation, a total of about 120 frames (one every 300 frames) in each sequence were labeled by hand; since images change smoothly from one frame to the next (apart from at scene changes), labeling more images would be unlikely to improve a learned model.

3.3 A Conditional Random Field Model of Object Classes

We use a conditional random field (CRF) model [Lafferty *et al.*, 2001] to learn the conditional distribution over the class labeling given an image. The use of a conditional random field allows us to incorporate shape, texture, color, location and edge cues in a single unified model. We define the conditional probability of the class labels $\mathbf{c}$ given an image $\mathbf{x}$ as

$$\begin{aligned}\log P(\mathbf{c}|\mathbf{x},\boldsymbol{\theta}) \quad = \quad & \sum_i \overbrace{\psi_i(c_i,\mathbf{x};\boldsymbol{\theta}_\psi)}^{\text{shape-texture}} + \overbrace{\pi(c_i,x_i;\boldsymbol{\theta}_\pi)}^{\text{color}} + \overbrace{\lambda(c_i,i;\boldsymbol{\theta}_\lambda)}^{\text{location}} \\ & + \sum_{(i,j)\in\mathcal{E}} \overbrace{\phi(c_i,c_j,\mathbf{g}_{ij}(\mathbf{x});\boldsymbol{\theta}_\phi)}^{\text{edge}} - \log Z(\boldsymbol{\theta},\mathbf{x}) \end{aligned} \tag{3.1}$$

where $\mathcal{E}$ is the set of edges in a 4-connected grid structure, $Z(\boldsymbol{\theta},\mathbf{x})$ is the partition function which normalizes the distribution, $\boldsymbol{\theta}=\{\boldsymbol{\theta}_\psi,\boldsymbol{\theta}_\pi,\boldsymbol{\theta}_\lambda,\boldsymbol{\theta}_\phi\}$ are the model parameters, and i and j index pixels in the image, which correspond to sites in the graph. Note that our model consists of three *unary* potentials which depend only on one node i in the graph, and one

pairwise potential depending on pairs of neighboring nodes in the graph. We next define the form of the four potential functions and their parameters.

Shape-texture potentials: The unary shape-texture potentials ψ use features selected by boosting to represent the shape, texture and appearance context of the object classes. These features and the boosting procedure used to perform feature selection while training a multi-class logistic classifier are described in detail in Section 3.4. We use this classifier directly as a potential in the CRF, so that

$$\psi_i(c_i, \mathbf{x}; \boldsymbol{\theta}_\psi) = \log P(c_i|\mathbf{x}, i) \tag{3.2}$$

where $P(c_i|\mathbf{x}, i)$ is the normalized distribution given by the boosted classifier, (3.16).

Color potentials: The unary color potentials capture the color distribution of instances of a class in a *particular image*. While the distribution of color across an entire class of objects is broad, the color distribution across one or a few instances of the class is relatively compact. Exploiting an accurate instance color model can therefore dramatically improve segmentation results [Blake *et al.*, 2004]. The parameters $\boldsymbol{\theta}_\pi$ must be learned separately for each image, and so this learning step needs to be carried out at test time. It is this potential that captures the more precise image-specific appearance that a solely class-specific recognition system cannot.

Our color models are represented as Gaussian Mixture Models (GMM) in CIELab color space where the mixture coefficients depend on the class label. The conditional probability of the color x of a pixel is given by

$$P(x|c) = \sum_k P(x|k)P(k|c) \tag{3.3}$$

with

$$P(x|k) = \mathcal{N}(x \mid \mu_k, \Sigma_k) \tag{3.4}$$

where k represents the mixture component the pixel is assigned to, and μ_k and Σ_k are the mean and variance respectively of mixture k. Notice that the mixture components are shared between different classes, and that only the coefficients $P(k|c)$ depend on the class label, making the model more efficient to learn than a separate GMM for each class. For a particular pixel x_i we compute a fixed soft assignment to the mixture com-

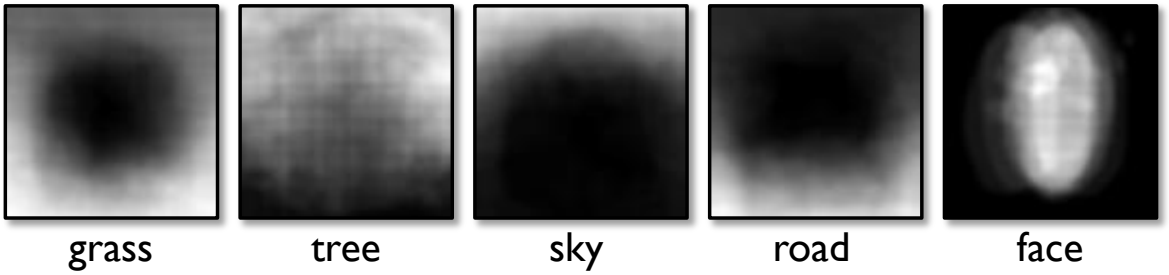

Figure 3.5: **Example location potentials.** Note how, for example, tree and sky pixels tend to occur in the upper half of images, while road pixels tends to occur at the bottom of the image. (White indicates increased probability).

ponents $P(k|x_i) \propto P(x_i|k)$ assuming a uniform prior $P(k)$.[1] Given this assignment, we specify our color potential to have the form

$$\pi(c_i, x_i; \boldsymbol{\theta}_\pi) = \log \sum_k \boldsymbol{\theta}_\pi(c_i, k) P(k|x_i) \tag{3.5}$$

where learned parameters $\boldsymbol{\theta}_\pi$ represent the distribution $P(c|k)$; note the conditional independence of c from x given k. (3.11) details the learning of $\boldsymbol{\theta}_\pi$.

Location potentials: The unary location potentials capture the (relatively weak) dependence of the class label on the absolute location of the pixel in the image. The potential takes the form of a look-up table with an entry for each class and pixel location:

$$\lambda_i(c_i, i; \boldsymbol{\theta}_\lambda) = \log \boldsymbol{\theta}_\lambda(c_i, \hat{i}) \; . \tag{3.6}$$

The index $\hat{i}$ is the normalized version of the pixel index i, where the normalization allows for images of different sizes: the image is mapped onto a canonical square and $\hat{i}$ indicates the pixel position within this square. Some learned location potentials are illustrated in Figure 3.5.

Edge potentials: The pairwise edge potentials ϕ have the form of a contrast sensitive Potts model [Boykov & Jolly, 2001],

$$\phi(c_i, c_j, \mathbf{g}_{ij}(\mathbf{x}); \boldsymbol{\theta}_\phi) = -\boldsymbol{\theta}_\phi^T \mathbf{g}_{ij}(\mathbf{x}) [c_i \neq c_j] \; , \tag{3.7}$$

[1] A soft assignment was found to give a marginal improvement over a hard assignment, at negligible extra cost.

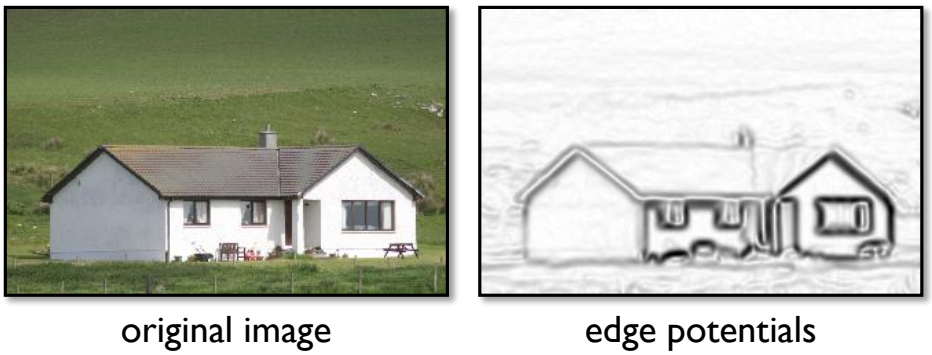

Figure 3.6: **Edge potentials for an example image.** The edge potentials in the CRF explicitly penalize neighboring nodes in the graph having different class labels, *except* where there is a corresponding edge in the image. Darker pixels in this image represent stronger edge responses and therefore lower costs.

with $[\cdot]$ the zero-one indicator function. In this work, the edge feature $\mathbf{g}_{ij}$ measures the difference in color between the neighboring pixels, as suggested by [Rother *et al.*, 2004],

$$\mathbf{g}_{ij} = \begin{bmatrix} \exp(-\beta\|x_i - x_j\|^2) \\ 1 \end{bmatrix} \tag{3.8}$$

where x_i and x_j are three-dimensional vectors representing the colors of pixels i and j respectively. Including the unit element allows a bias to be learned, to remove small, isolated regions. The quantity β is an image-dependent contrast term, and is set separately for each image to $(2\langle\|x_i - x_j\|^2\rangle)^{-1}$, where $\langle\cdot\rangle$ denotes an average over the image. An example using the function $\exp(-\beta\|x_i - x_j\|^2)$ is shown in Figure 3.6.

3.3.1 Learning the CRF: MAP Training

Ideally, the parameters of the model should be learned with maximum likelihood (ML) estimation [Kumar & Hebert, 2003] or, better, maximum *aposteriori* (MAP) learning: these methods maximize the conditional likelihood of the labels given the training data, and for MAP learning incorporate a prior term to prevent over-fitting:

$$L(\boldsymbol{\theta}) = \sum_n \log P(\mathbf{c}_n|\mathbf{x}_n, \boldsymbol{\theta}) + P(\boldsymbol{\theta}) \; . \tag{3.9}$$

The maximization of $L(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ can be achieved using a gradient ascent algorithm, iteratively computing the gradient of the conditional likelihood with respect to each parameter, $\frac{\partial}{\partial\theta_i}L(\boldsymbol{\theta})$, and moving up the gradient. This requires the evaluation of marginal

probabilities over the class labels at each pixel for all training images.

Exact computation of these marginals is sadly infeasible, since it would require vast computational effort to perform the numerous marginalizations of (3.1). Instead, we approximated the label marginals by the mode, the most probable labeling, inferred using alpha-expansion graph cuts as described in Section 3.3.3. This approximation was made because the size of our datasets limited the time available to estimate marginals; alternative, slower but more accurate approximations such as loopy belief propagation (BP) [Pearl, 1988; Yedidia *et al.*, 2003] or variational methods [Beal, 2003] could also be investigated.

We attempted MAP learning of the several thousand shape-texture potential parameters $\boldsymbol{\theta}_\psi$ and the two edge potential parameters $\boldsymbol{\theta}_\phi$. For the $\boldsymbol{\theta}_\psi$, the optimization was performed over the a and b parameters of the weak learners in (3.17), initialized at the values given by boosting.

However, the modal approximation used proved insufficient for estimating such a large number of parameters. Conjugate gradient ascent did eventually converge to a solution, but evaluating the learned parameters against validation data gave poor results with almost no improvement on unary classification alone. Additionally, for the learning of the edge potential parameters, the lack of alignment between object edges and label boundaries in the roughly labeled training set forced the learned parameters to tend toward zero.

3.3.2 Learning the CRF: Piecewise Training

Given these problems with directly maximizing the conditional likelihood, we decided instead to use a method based on *piecewise training* [Sutton & McCallum, 2005]. Piecewise training involves dividing the CRF model into pieces corresponding to the different terms in (3.1). Each of these pieces is then trained independently, as if it were the only term in the conditional model. For example, if we apply piecewise training to the CRF model of Figure 3.7(a), the parameter vectors $\boldsymbol{\theta}_\phi$, $\boldsymbol{\theta}_\psi$ and $\boldsymbol{\theta}_\pi$ are learned by maximizing the conditional likelihood in each of the three models of Figure 3.7(b). In each case, only the factors in the model that contain the relevant parameter vector are retained.

As discussed in [Sutton & McCallum, 2005], this training method minimizes an upper bound on the log partition function, as follows: if we define the logarithm of the partition function $z(\boldsymbol{\theta}, \mathbf{x}) = \log Z(\boldsymbol{\theta}, \mathbf{x})$ and index the terms in the model by r, then

$$z(\boldsymbol{\theta}, \mathbf{x}) \leq \sum_r z_r(\boldsymbol{\theta}_r, \mathbf{x}) \tag{3.10}$$

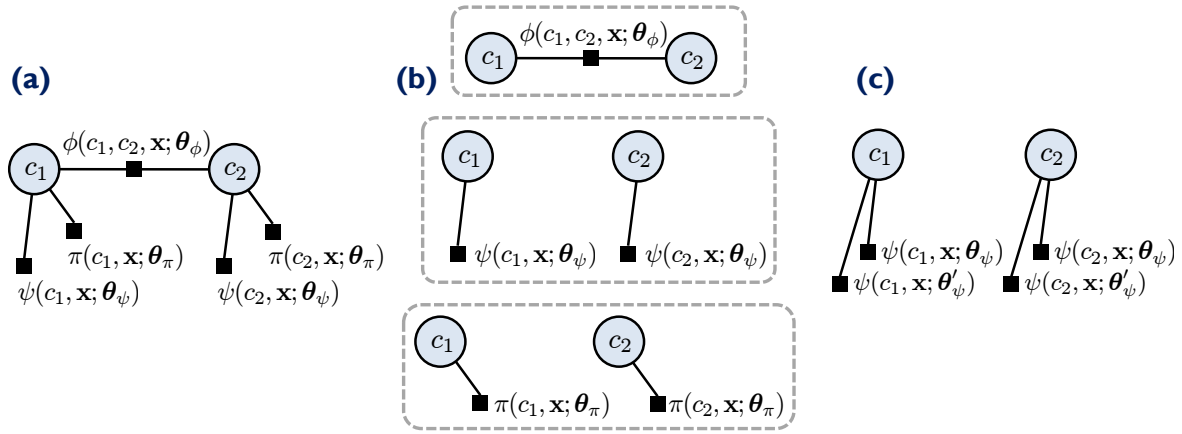

Figure 3.7: **Piecewise training of the CRF parameters. (a)** The factor graph (see e.g. [Bishop, 2006]) for a simplified CRF model. Each black square represents a term in (3.1) and each circle represents a latent variable. Terms are connected to all variables that they depend on. **(b)** When piecewise training is applied to the CRF model of **(a)**, the parameters $\boldsymbol{\theta}_\phi$, $\boldsymbol{\theta}_\psi$ and $\boldsymbol{\theta}_\pi$ are learned by maximizing conditional likelihood in the top, middle and bottom models respectively. In each model, only the terms relating to the parameter being learned are retained. **(c)** A model in which the term ψ has been duplicated. In this case, piecewise training will learn model parameters which are twice those learned in the original non-duplicated model. Hence, piecewise training will lead to over-counting errors when terms in the model are correlated. See text for more details.

where $\boldsymbol{\theta}_r$ are the parameters of the rth term and $z_r(\boldsymbol{\theta}_r)$ is the partition function for a model containing only the rth term. Replacing $z(\boldsymbol{\theta}, \mathbf{x})$ with $\sum_r z_r(\boldsymbol{\theta}_r)$ in (3.1) then gives a lower bound on the conditional likelihood. It is this bound which is maximized during piecewise learning.

Unfortunately, this bound can be loose, especially if the terms in the model are correlated. In this case, performing piecewise parameter training leads to over-counting during inference in the combined model. To understand this, consider the case where we duplicate a term of the model $\psi(\boldsymbol{\theta}_\psi)$, so that there is an additional term $\psi(\boldsymbol{\theta}'_\psi)$ which has the same functional form but new parameters $\boldsymbol{\theta}'_\psi$. A model with duplicated terms is shown in Figure 3.7(c). As the duplicate terms have the same form and are based on the same features, these terms are perfectly correlated.

Piecewise training will learn that $\boldsymbol{\theta}_\psi$ and $\boldsymbol{\theta}'_\psi$ are the same and equal to the parameters $\boldsymbol{\theta}_\psi^{\text{old}}$ learned for this term in the original model. Since the log potential function $\log \psi$ is linear in the parameters, the duplicate model will be equivalent to the original model but with $\boldsymbol{\theta}_\psi^{\text{new}} = 2\boldsymbol{\theta}_\psi^{\text{old}}$, i.e. twice the correct value. To offset this over-counting effect and recover the original parameters, it would be necessary to weight the logarithm of each duplicate term by a factor of 0.5, or equivalently raise the term to the power of 0.5.

It is difficult to assess analytically the degree of over-counting introduced by dependencies between the different terms in our CRF model. Instead, we introduce scalar powers for each term and optimize these powers discriminatively using cross-validation on a set of validation images. We found that it was only necessary to introduce powers for the location and color potentials. It can be shown that this leads to an approximate partition function of similar form to (3.10), except that it is no longer an upper bound on the true partition function.

Piecewise training with powers is therefore used to train the parameters of each of the potential types separately as follows.

Shape-texture potential parameters: The shape-texture potential parameters are learned during boosting, described in Section 3.4.

Color potential parameters: At test time the color potential parameters are learned for each image in a piecewise fashion, similarly to [Rother *et al.*, 2004]. First a class labeling $\mathbf{c}^\star$ is inferred (see Section 3.3.3) and then the color parameters are updated using

$$\boldsymbol{\theta}_\pi(c_i, k) = \left(\frac{\sum_i [c_i = c_i^\star] P(k|x_i) + \alpha_\pi}{\sum_i P(k|x_i) + \alpha_\pi} \right)^{w_\pi} . \tag{3.11}$$

Given this new parameter setting, a new class labeling is inferred and this procedure is iterated. In practice, the Dirichlet prior parameter α_π was set to 0.1, the power parameter was set as $w_\pi = 3$, and fifteen color components and two iterations gave good results. Because we are training in pieces, the color parameters do not need to be learned for the training set.

Location potential parameters: We train the location potential parameters by maximizing the likelihood of the normalized model containing just that potential and raising the result to a fixed power w_λ to compensate for over-counting. This corresponds to

$$\boldsymbol{\theta}_\lambda(c, \hat{i}) = \left(\frac{N_{c,\hat{i}} + \alpha_\lambda}{N_{\hat{i}} + \alpha_\lambda} \right)^{w_\lambda} \tag{3.12}$$

where $N_{c,\hat{i}}$ is the number of pixels of class c at normalized location $\hat{i}$ in the training set, $N_{\hat{i}}$ is the total number of pixels at location $\hat{i}$ and α_λ is a small integer (we use $\alpha_\lambda = 1$) corresponding to a weak Dirichlet prior on $\boldsymbol{\theta}_\lambda$. The parameter w_λ was changed between the different datasets; the relevant values are given in Section 3.5.

Edge potential parameters: The values of the two contrast-related parameters were manually selected to minimize the error on the validation set.

3.3.3 Inference in the CRF Model

Given a set of parameters learned for the CRF model, we wish to find the most probable labeling $\mathbf{c}^\star$, i.e. the labeling that maximizes the conditional probability of (3.1). The optimal labeling is found by applying the alpha-expansion graph-cut algorithm of [Boykov & Jolly, 2001]. Note that the energy is alpha-expansion *submodular* (this term has superseded the original term *regular*; see [Kolmogorov & Zabih, 2004] for details).

The idea of the expansion move algorithm is to reduce the problem of maximizing a function $f(\mathbf{c})$ (corresponding to (3.1)) with *multiple* labels to a sequence of *binary* maximization problems. These sub-problems are called *alpha-expansions*. They can be described as follows (see [Boykov & Jolly, 2001] for details). Suppose that we have a current configuration (set of labels) $\mathbf{c}$ and a fixed label $\alpha \in \{1, \ldots, C\}$, where C is the number of classes. In the alpha-expansion operation, each pixel i makes a binary decision: it can either keep its old label or switch to label α. Therefore, we introduce a binary vector $\mathbf{s} \in \{0,1\}^{\mathcal{P}}$ which defines the auxiliary configuration $\mathbf{c}[\mathbf{s}]$ as follows:

$$c_i[\mathbf{s}] = \begin{cases} c_i & \text{if } s_i = 0 \\ \alpha & \text{if } s_i = 1. \end{cases} \tag{3.13}$$

This auxiliary configuration $\mathbf{c}[\mathbf{s}]$ transforms the function f with *multiple* labels into a function of *binary* variables $f'(\mathbf{s}) = f(\mathbf{c}[\mathbf{s}])$. Because our edge potentials are attractive, the global maximum of this binary function can be found exactly using graph cuts.

The expansion move algorithm starts with an initial configuration $\mathbf{c}^0$.[2] It then computes optimal alpha-expansion moves for labels α in some order, accepting the moves only if they increase the objective function. The algorithm is guaranteed to converge and its output is a strong local maximum, characterized by the property that no further α-expansion can increase the function f.

[2] In our case, the initial configuration $\mathbf{c}^0$ for the alpha-expansion is given by the mode of the shape-texture potentials, though the final MAP solution was not in practice sensitive to this. Additionally, the order of expansion moves did not have a noticeable effect on performance.

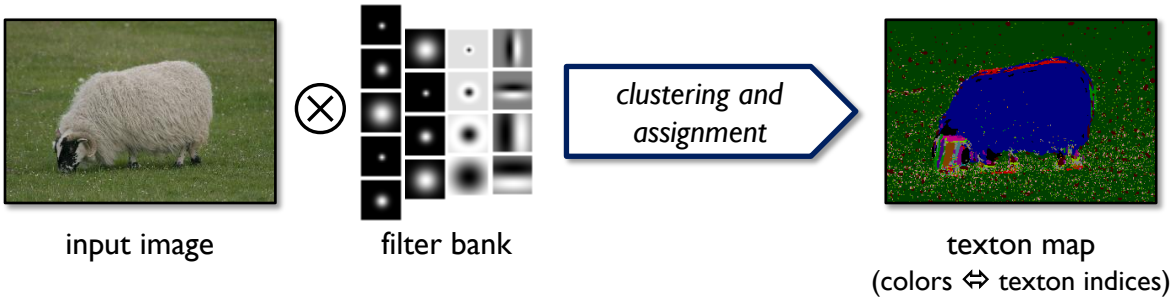

Figure 3.8: **The process of image textonization.** An input image is convolved with a filter bank. The filter responses for all pixels in training images are clustered. Finally each pixel is assigned a texton index corresponding to the nearest cluster center to its filter responses.

3.4 Boosted Learning of Shape, Texture and Context

The most important term in the CRF energy is the unary shape-texture potential of (3.2), which is based on a novel set of features which we call *shape filters*. These features are capable of capturing shape, texture and appearance context jointly. In this section, we describe shape filters and the boosting process for automatic feature selection and learning of the shape-texture potentials.

3.4.1 Textons

Efficiency demands a compact representation for the range of different appearances of an object. For this we use *textons* [Leung & Malik, 2001] which have been proven effective in categorizing materials [Varma & Zisserman, 2005] as well as generic object classes [Winn *et al.*, 2005]. The term texton was coined by [Julesz, 1981] for describing human textural perception, and is somewhat analogous to phonemes used in speech recognition.

The training images are convolved with a 17-dimensional filter bank at scale κ.[3] The 17-D responses for all training pixels are then whitened (to give zero mean and unit covariance), and an unsupervised clustering is performed. We employ the Euclidean-distance K-means clustering algorithm, which can be made dramatically faster by using the techniques of [Elkan, 2003]. Finally, each pixel in each image is assigned to the nearest cluster

[3]The choice of filter bank is somewhat arbitrary, as long as it is sufficiently representative. We use the same filter bank as [Winn *et al.*, 2005], which consists of Gaussians at scales κ, 2κ and 4κ, x and y derivatives of Gaussians at scales 2κ and 4κ, and Laplacians of Gaussians at scales κ, 2κ, 4κ and 8κ. The Gaussians are applied to all three color channels, while the other filters are applied only to the luminance. The perceptually uniform CIELab color space is used.

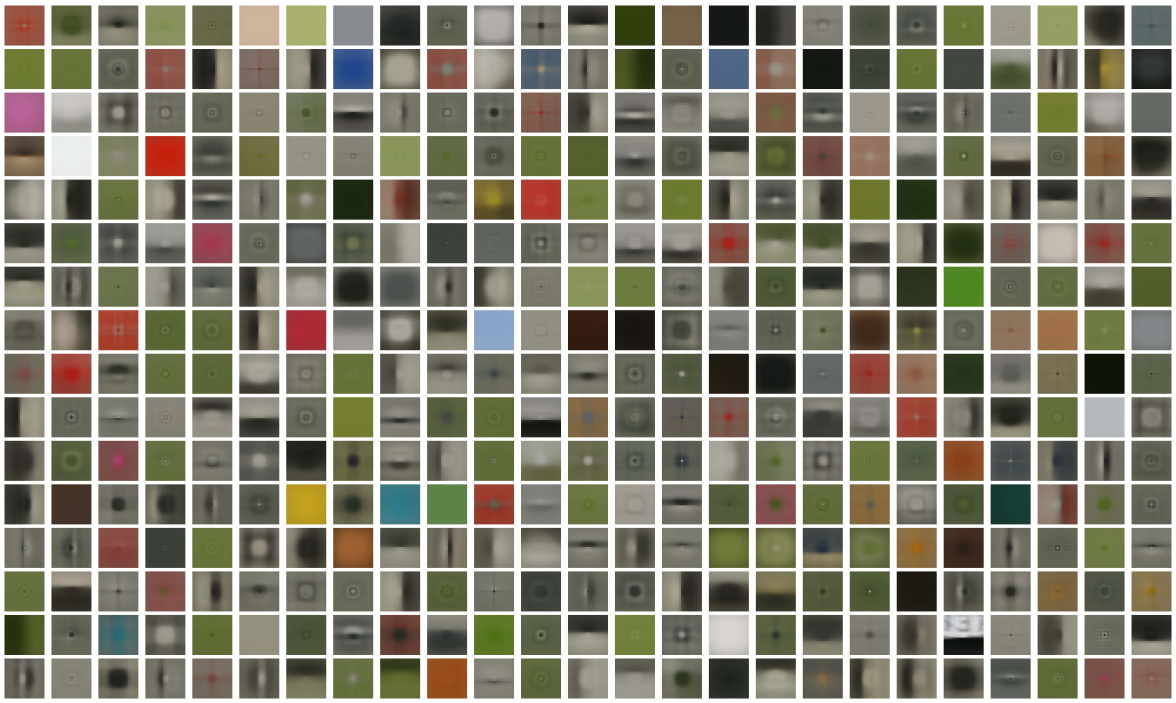

Figure 3.9: **Textons for the MSRC 21 class dataset.** 400 patches corresponding to the $K = 400$ learned textons are shown. The patch corresponding to texton t shows the average across the dataset of 25x25 pixel windows centered on all pixels i with $T_i = t$. Observe structures corresponding to untextured regions of constant color, horizontal and vertical bars and edges (including some colored edges), blobs, rings, and crosses. Due to the choice of filter bank, no diagonal edges are present, although the learning algorithm is to some extent able to compensate for such missing types of texture.

center, producing the *texton map*. We will denote the texton map as T where the pixel i has value $T_i \in \{1, \ldots, K\}$.

This process of textonization is illustrated in Figure 3.8, and a visualization of the resulting textons is shown in Figure 3.9. Additionally, a simplistic reconstruction, termed *inverse textonization*, in which the average texton patches from Figure 3.9 are superimposed and averaged, is illustrated in Figure 3.10.

Note that for efficiency one can use the k-d tree algorithm [Beis & Lowe, 1997] to perform the nearest neighbor search; without any approximation, textonization using k-d trees with leaf-node bins containing 30 cluster centers gave a speed up of about 5 times over simple linear search.

3.4.2 Shape Filters

Each shape filter is a pair (r, t) of an arbitrary shape, r, and a texton t. In our implementation, the allowed shapes are rectangular regions, in coordinates relative to a pixel i being

Figure 3.10: **Inverse textonization.** Given the texton map and the average texton patches from Figure 3.9, a good approximation to the original image can be reconstructed. This highlights the representative power of textons.

classified. For simplicity, a set of N_{R} candidate rectangles are chosen at random, such that their top-left and bottom-right corners lie within a fixed bounding box covering about half the image area.[4] The feature response at location i is the count of instances of texton t under the offset rectangle mask $r + i$

$$v_{[r,t]}(i) = \sum_{j \in (r+i)} [T_j = t] \,. \tag{3.14}$$

Outside the image boundary there is zero contribution to the feature response. Figure 3.11 illustrates this process.

The filter responses can be efficiently computed over a whole image with integral images [Viola & Jones, 2001; Porikli, 2005]. Figure 3.12 illustrates this process: the texton map is separated into K channels (one for each texton) and then, for each channel, a separate integral image is calculated. These can later be used to compute the shape filter responses in constant time: if $\hat{T}^{(t)}$ is the integral image of T for texton channel t, then the feature response is computed as:

$$v_{[r,t]}(i) = \hat{T}^{(t)}_{r_{\mathrm{br}}} - \hat{T}^{(t)}_{r_{\mathrm{bl}}} - \hat{T}^{(t)}_{r_{\mathrm{tr}}} + \hat{T}^{(t)}_{r_{\mathrm{tl}}} \tag{3.15}$$

[4]For the evaluations in this chapter, the bounding box was ± 100 pixels in x and y. This allows the model to exploit appearance context over a long range, despite the CRF having connections only to pixel-wise neighbors. The CRF is still very important however: it allows us additionally to exploit the edge, color, and location potentials to achieve near pixel-perfect segmentation.

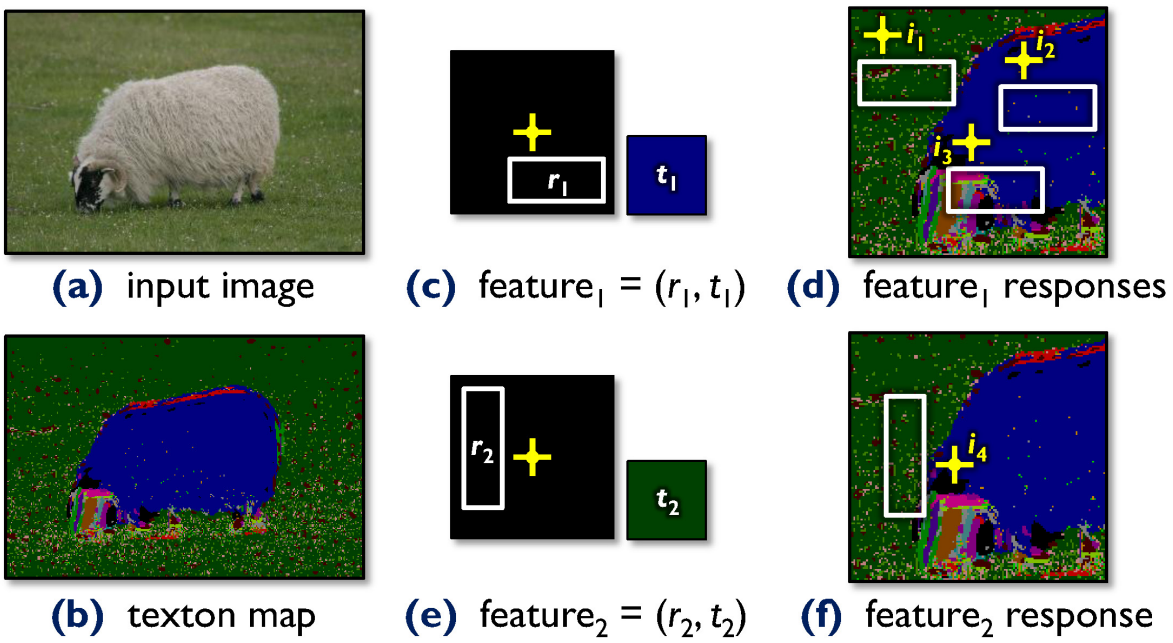

Figure 3.11: **Calculating feature responses and capturing appearance context. (a, b)** An image and its corresponding texton map (colors map uniquely to texton indices). **(c)** A rectangle mask r_1 (white) is defined relative to the yellow cross which represents the point i being classified, and paired with texton t_1 which here maps to the blue color. **(d)** As an example, the feature response $v_{[r_1,t_1]}(i)$ is calculated at three positions in the texton map (zoomed). If A is the area of r, then in this example $v_{[r_1,t_1]}(i_1) \approx 0$, $v_{[r_1,t_1]}(i_2) \approx A$, and $v_{[r_1,t_1]}(i_3) \approx A/2$. **(e)** A second feature with a different rectangle mask r_2 is paired with texton t_2 which maps to the green color. **(f)** For this feature where t_2 corresponds to grass, our algorithm learns that points i (such as i_4) belonging to sheep regions tend to produce large counts $v_{[r_2,t_2]}(i)$, and hence exploits the contextual information that sheep pixels tend to be surrounded by grass pixels.

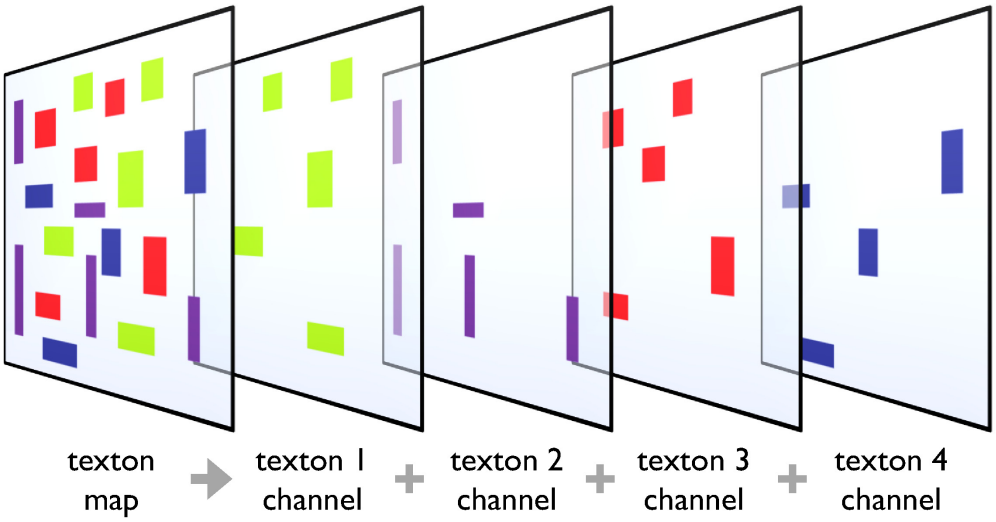

Figure 3.12: **Separating the texton map into multiple channels.** The texton map of an image, containing K textons, is split into K channels. An integral image is built for each channel and used to compute shape filter responses in constant time.

where r_{br}, r_{bl}, r_{tr} and r_{tl} denote the bottom right, bottom left, top right and top left corners of rectangle r.

Shape filters, as pairs of rectangular masks and textons, can be seen as an extension of the features used in [Viola & Jones, 2001]. Our features are sufficiently general to allow us to *learn* automatically shape and context information, in contrast to techniques such as shape context [Belongie *et al.*, 2002] which utilize a hand-picked shape descriptor. Figure 3.11 illustrates how shape filters are able to model appearance-based context, and a toy example in Figure 3.13 demonstrates how shape filters model shape and layout.

Variations on Shape Filters

A further *appearance-independent* shape filter can also be used to model just shape. This special feature acts exactly like a normal shape filter, except that it does not have a particular texton to which it is applied, but rather uses whichever texton is at the pixel being classified, i.e. the feature response calculated is $v_{[r,T_i]}(i)$. The addition of this appearance-independent shape filter is evaluated in Section 3.5.3.

Additionally, we investigated other types of shape beyond simple rectangles. In particular we evaluated rectangles rotated by $45°$, and pairs of rectangles with the texton responses either added ($v_{[r_1,t]}(i) + v_{[r_2,t]}(i)$) or subtracted ($v_{[r_1,t]}(i) - v_{[r_2,t]}(i)$). However, despite considerable extra computational expense (since these new combinations of features must be tested at each round of boosting; see below), the more complicated features did not produce

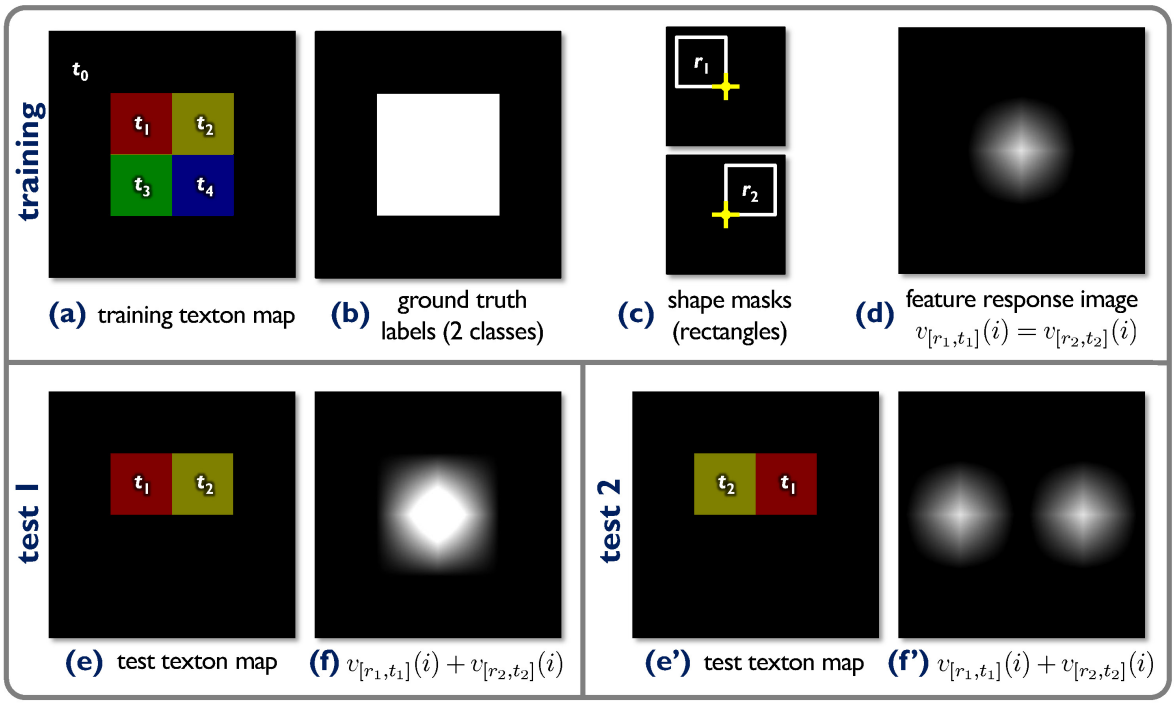

Figure 3.13: **Capturing local shape information.** This toy example illustrates how our *shape filters* capture relative positions of textons. **(a)** Input texton map. **(b)** Input binary ground truth label map (foreground=white, background=black). **(c)** Example rectangle masks (r_1 and r_2). **(d)** The feature response image $v_{[r_1,t_1]}(i)$ shows a positive response within the foreground region and zero in the background. An identical response image is computed for feature (r_2, t_2). Boosting would pick both these features as discriminative. **(e)** A test input with textons t_1 and t_2 in the *same* relative position as that of training. **(f)** Illustration that the two feature responses *reinforce* each other. **(e')** A second test with t_1 and t_2 swapped. **(f')** The summed feature responses do not reinforce, giving a weaker signal for classification. Note that **(f)** and **(f')** are illustrative only, since boosting actually combines thresholded feature responses, though the principle still applies.

noticeably better results. We believe this to be due to over-fitting.

We have also tried modeling appearance using the learned visual words of [Winn *et al.*, 2005]. Unsurprisingly, the classification results were worse than using the raw K-means clusters, since the learning algorithm in [Winn *et al.*, 2005] performs clustering so as to be invariant to the spatial layout of the textons – exactly the opposite of what is required here.

3.4.3 Learning Shape Filters using Joint Boost

We employ an adapted version of the Joint Boost algorithm of [Torralba *et al.*, 2004] to select discriminative shape filters, while simultaneously learning the shape-texture potentials. This multi-class extension of the Gentle AdaBoost algorithm (used in Chapter 2) is detailed in Section B.3. The algorithm iteratively builds an accurate, strong classifier as a sum of weak learners, while simultaneously selecting discriminative features. Each weak learner is a decision stump based on a weighted, thresholded feature response $v_{[r,t]}(i)$, and is *shared* between a set of classes $\mathcal{C}$, allowing a single feature to support the classification of several classes at once. This sharing of features over classes allows for classification with cost sub-linear in the number of classes, and leads to improved generalization; see [Torralba *et al.*, 2004].

The learned strong classifier is an additive model of the form $H(c,i) = \sum_{m=1}^{M} h_m(c,i)$, summing the classification confidence of M weak classifiers. This confidence value can be reinterpreted as a probability distribution over c using the soft-max or multi-class logistic transformation [Friedman *et al.*, 2000] to give the shape-texture potentials:

$$P(c|\mathbf{x},i) \propto \exp H(c,i) \ . \tag{3.16}$$

Each weak learner is a decision stump of the form

$$h(c,i) = \begin{cases} a[v_{[r,t]}(i) > \theta] + b & \text{if } c \in \mathcal{C} \\ k^c & \text{otherwise,} \end{cases} \tag{3.17}$$

with parameters $(a, b, \{k^c\}_{c \notin \mathcal{C}}, \theta, \mathcal{C}, r, t)$. The r and t indices together specify the shape filter feature (rectangle mask and texton respectively), with $v_{[r,t]}(i)$ representing the corresponding feature response at position i. For those classes that share this feature ($c \in \mathcal{C}$), the weak learner gives $h(c,i) \in \{a+b, b\}$ depending on the comparison of $v_{[r,t]}(i)$ to a threshold θ. For each class not sharing the feature ($c \notin \mathcal{C}$), the constant k^c ensures that unequal numbers of

Figure 3.14: **Boosting as feature selection.** The Joint Boost algorithm acts on a matrix of shape filter responses $v_{[r,t]}(i)$, and iteratively picks out a feature dimension corresponding to a particular shape filter. The algorithm also selects a set of classes $\mathcal{C}$ between which the chosen shape filter is shared. In this illustration, after two rounds, two shape filters have been selected from the pool of six. For the datasets considered in this work, thousands of shape filters are selected from a pool of hundreds of thousands.

training examples of each class do not adversely affect the learning procedure.

As illustrated in Figure 3.14, the boosting algorithm is given a matrix of responses $v_{[r,t]}(i)$, calculated for all N_{R} candidate shapes and all textons t at a number of example image locations i. Additionally, a target class $z_i \in \{1, \ldots, C\}$ is provided from the ground truth labeling. Each weak learner (3.17) selected by boosting corresponds directly to a shape filter (r, t), and hence, over a number of rounds, boosting builds the strong classifier $H(c, i)$ and thereby the shape-texture potentials $P(c|\mathbf{x}, i)$, by combining many shape filters. Note that in practice, the matrix of feature responses is too large to fit entirely in memory and is efficiently computed on-the-fly.

We conclude our discussion of the learning algorithm with important comments on efficiency.

Sub-sampling: The considerable memory and processing requirements of this procedure make training with an example at every pixel impractical. Hence we take examples only at pixels lying on a $\Delta_{\mathrm{ss}} \times \Delta_{\mathrm{ss}}$ grid ($\Delta_{\mathrm{ss}} = 3$ for the Corel and Sowerby datasets, which contain smaller images, and $\Delta_{\mathrm{ss}} = 5$ for the other datasets with larger images). The shape filter responses are still calculated at full resolution to allow for per-pixel accurate classification at test time; we simply calculate and store the responses at fewer image locations.

One consequence of this sub-sampling is that a small degree of shift-invariance is learned. On its own, this might lead to inaccurate segmentation at object boundaries. However, when applied in the context of the CRF, the edge and color potentials come

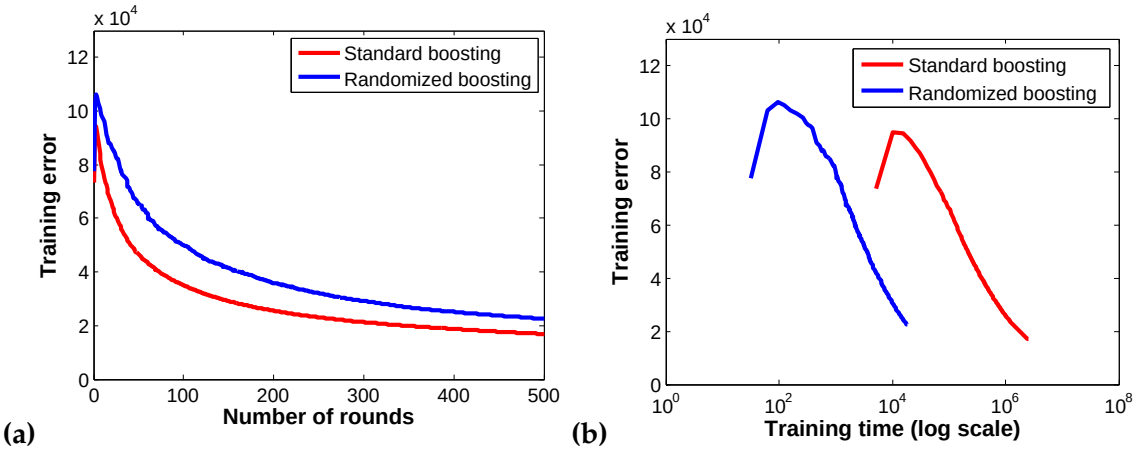

Figure 3.15: **Effect of random feature selection on a toy example. (a)** Training error as a function of the number of rounds (axis scales are unimportant). **(b)** Training error as a function of time. Randomization makes learning two orders of magnitude faster here, with very little increase in training error for the same number of rounds. The initial peak in error is due to the imbalance in the number of training examples for each class; on the log scale this appears quite significant, but in fact it affects at most the first five rounds of boosting.

into effect to locate the object boundary accurately.

Random feature selection: Even when using sub-sampling, exhaustive search over all features (r, t) at each round of boosting is prohibitive. For efficiency therefore, our algorithm examines only a randomly chosen fraction $\zeta \ll 1$ of the possible features (see [Baluja & Rowley, 2005]). All results in this chapter use $\zeta = 0.003$ so that, over several thousand rounds, there is high probability of testing all features at least once, and hence good features should eventually get selected.

To analyze the effect of random feature selection, we compared the results of boosting on a toy data set of ten images with ten rectangle masks, and 400 textons. The results in Figure 3.15 show that random feature selection improves the training time by two orders of magnitude whilst having only a small impact on the training error. Additionally, although we have no formal experiments to prove this, our experience with randomization has been that decreasing ζ occasionally gives an overall gain in test performance. This perhaps suggests that randomization is not only speeding up learning, but also improving generalization by preventing over-fitting to the training data.

Forests of boosted classifiers: A further possible efficiency, though not evaluated here, is the use of a forest of TextonBoost classifiers. In a similar manner to [Lepetit *et al.*,

2005], several shape-texture potential classifiers can be trained on random subsets of the image data and combined by averaging:

$$P(c|\mathbf{x}, i) = \frac{1}{W} \sum_{w=1}^{W} P^{[w]}(c|\mathbf{x}, i) \ . \tag{3.18}$$

This allows infeasibly large datasets to be partitioned into smaller, manageable subsets, and has the potential to improve the generalization ability of the shape-texture potentials.

3.4.4 Separable Shape Filters

We propose a final variation for speed critical applications, e.g. processing video sequences or large images. Here, two TextonBoost classifiers are learned to act on the two separate Cartesian axes. A horizontal classifier $P(\mathbf{c}_{\mathrm{x}}|\mathbf{x}, \boldsymbol{\theta}_{\mathrm{x}})$, representing the class probabilities for each column, and a vertical classifier $P(\mathbf{c}_{\mathrm{y}}|\mathbf{x}, \boldsymbol{\theta}_{\mathrm{y}})$, representing the class probabilities for each row, can be combined as the outer product

$$P(\mathbf{c}|\mathbf{x}, \boldsymbol{\theta}) \approx P(\mathbf{c}_{\mathrm{x}}|\mathbf{x}, \boldsymbol{\theta}_{\mathrm{x}}) \times P(\mathbf{c}_{\mathrm{y}}|\mathbf{x}, \boldsymbol{\theta}_{\mathrm{y}}) \ . \tag{3.19}$$

This factorized approximation is clearly less powerful than learning the full joint classifier, but as shown in Section 3.5.4, gives acceptable quantitative performance and a considerable speed-up.

We investigate these 1D classifiers using the shape-texture potentials without the other terms in the CRF. As illustrated in Figure 3.16, separable shape filters use spans instead of rectangles: horizontal spans count the number of textons agreeing in texton index that lie in a horizontal strip relative to the y coordinate being classified; vertical spans do similarly for a vertical strip. The target values for training the separable classifiers become the *set* of all classes present in column x or row y.

3.5 Results and Comparisons

In this section we investigate the performance of our semantic segmentation algorithm on several challenging datasets, and compare our results with existing work. We first investigate the effect of different aspects of the model, and then show the full quantitative and qualitative results.

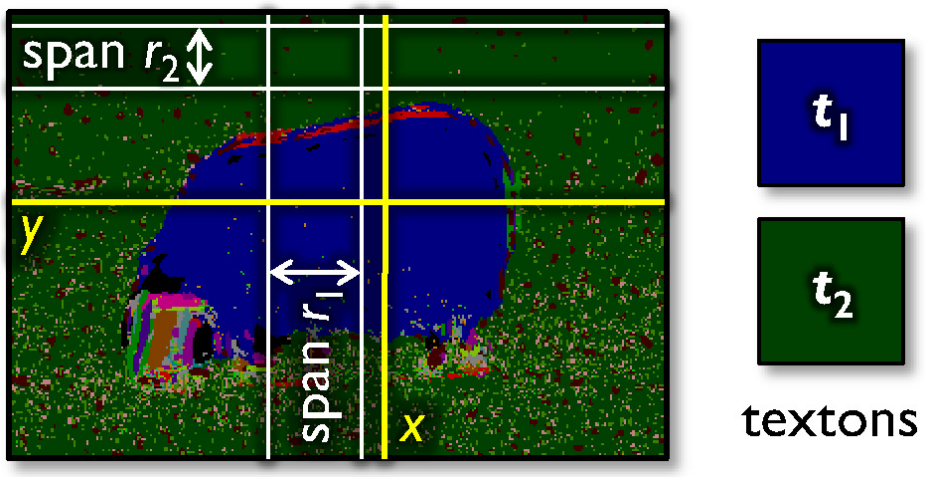

Figure 3.16: **Separable shape filters.** For speed, separable shape filters may be used. Horizontal spans are defined relative to the y coordinate being classified, and vertical spans relative to the x coordinate. The response of separable shape filter (r, t) is computed as the count of pixels within the span r that have texton index t (cf. Figure 3.11). The classifiers for the two separate axes are combined as (3.19). In this example, separable shape filter (r_1, t_1) uses the presence of texton t_1 in span r_1 as evidence that sheep is present at coordinate x. Feature (r_2, t_2) exploits appearance context, looking for regions of grass texton t_2 in span r_2 above the sheep at coordinate y.

3.5.1 Boosting Accuracy

In Figure 3.17 we illustrate how boosting gradually selects new shape filters to improve classification accuracy. Initially, after 30 rounds of boosting corresponding to 30 shape filters, a very poor classification is given, with low confidence. As more shape filters are added, the classification accuracy improves greatly, and after 2000 rounds a very accurate classification is given. Note that this illustrates only the shape-texture potentials, and not the full CRF model.

Figure 3.18(a) illustrates the effect of training the shape-texture potentials using boosting on the MSRC dataset. As expected, the training error J_{wse} (B.13) decreases non-linearly as the number of weak classifiers increases. Furthermore, Figure 3.18(b) shows the accuracy of classification with respect to the validation set, which after about 5000 rounds flattens out to a value of approximately 73%. The accuracy against the validation set is measured as the pixel-wise segmentation accuracy, in other words the percentage of pixels that are assigned the correct class label.

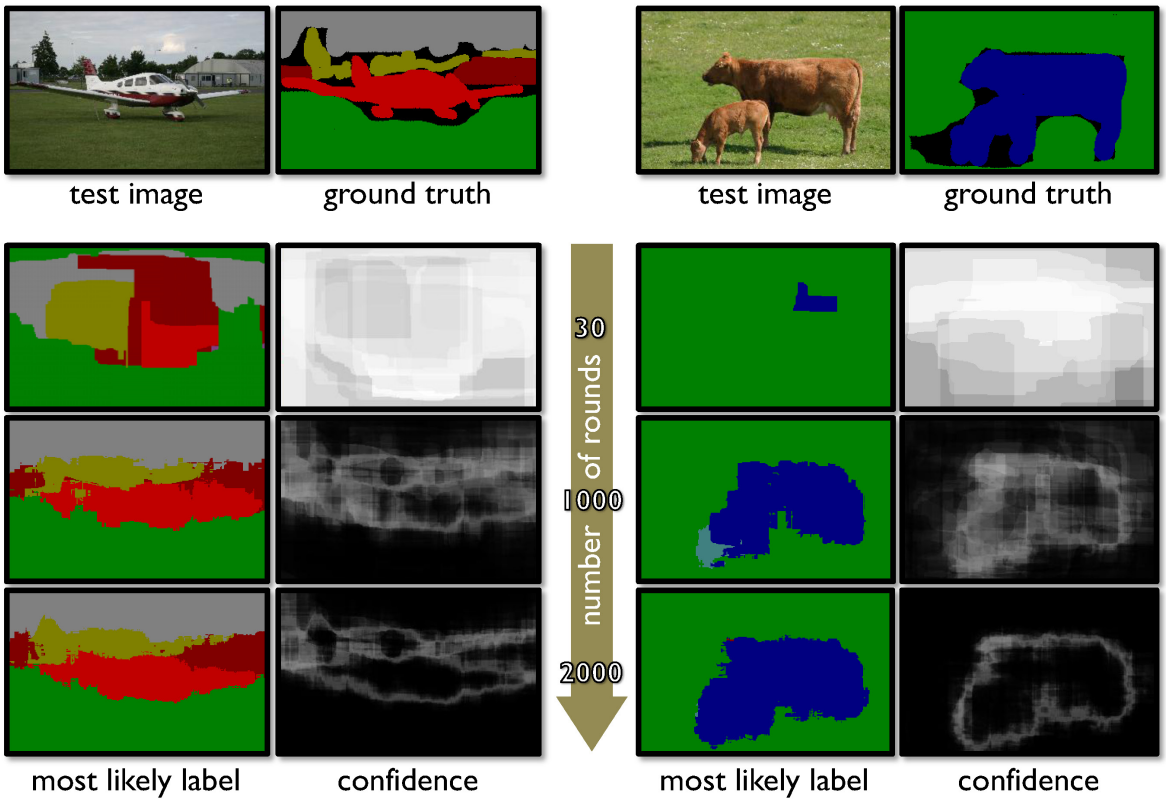

Figure 3.17: **More shape filters improve classification. Top row**: Unseen test images and the corresponding ground truth. **Lower three rows**: classification output of the shape-texture potentials trained by boosting, as more shape filters are used. The three rows show the most likely label maps and the confidence maps with 30, 1000 and 2000 weak learners. Colors represent class labels (see Figure 3.21 for color key). White represents low confidence, black high confidence. Confidence is computed as the entropy of the class label distribution at each point.

3.5.2 The Effect of Different Model Potentials

Figure 3.19 shows results for variations of our model with different potentials included. It is evident that imposing spatial coherency (c) as well as an image dependent color model (d) improves the results considerably.

The percentage accuracies in Figure 3.19 show that each term in our model captures essential information from the training set. Note that the improvement given by the full model over just the shape-texture potentials, while numerically small, corresponds to a significant increase in perceived accuracy (compare (b) with (d) in Figure 3.19), since the object contour is more accurately delineated.

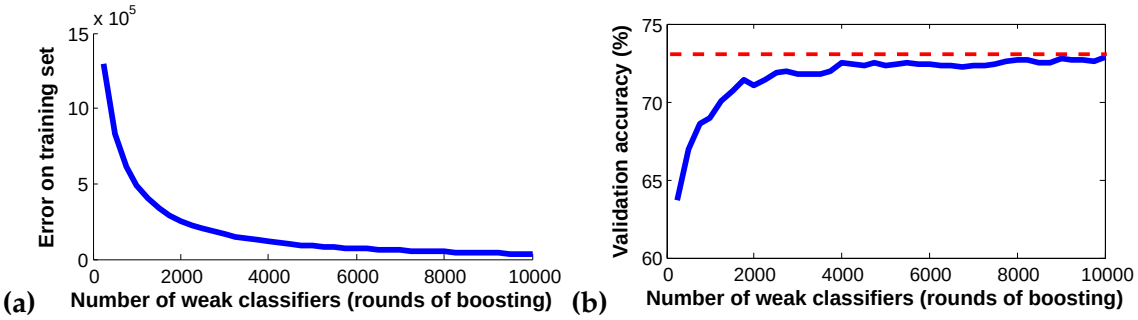

Figure 3.18: **Training and validation error.** Training error **(a)** and accuracy on the validation set **(b)** as functions of the number of weak classifiers. While the training error decreases almost to zero, the validation set accuracy rises to a maximum of about 73%. Validation accuracy values given are pixel-wise segmentation accuracies.

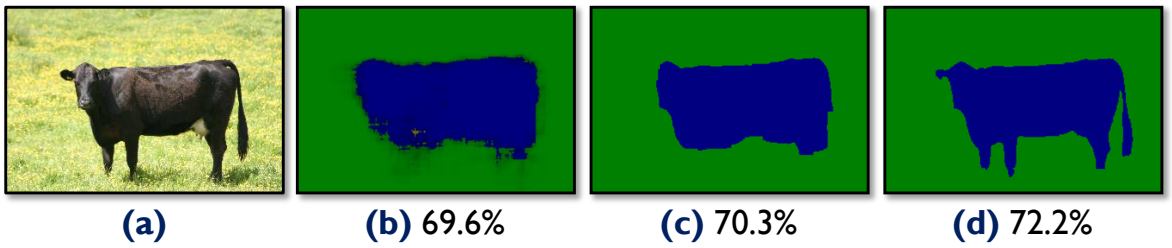

Figure 3.19: **Effect of different model potentials**. The original input image **(a)** and the result from the boosted classifier alone **(b)**, with no explicit spatial coherency; at the boundary between blue and green, darker pixels correspond to higher entropy of the unary potentials. **(c)** Results for the CRF model without color modeling, i.e. omitting term π in (3.1), and **(d)** for the full CRF model. Segmentation accuracy figures are given over the whole dataset. Observe the marked improvement in perceived segmentation accuracy of the full model over the boosted classifier alone, despite a seemingly small numerical improvement.

3.5.3 Appearance-Independent Shape Filters

As described in Section 3.4.2, we investigated the use of appearance-independent shape filters, that used the texton at the pixel being classified to calculate the feature responses. An experiment was performed on the Corel dataset with 1000 rounds of boosting, $\kappa = 0.45$, and using just the shape-texture potentials. We compared the performance on the test set in terms of pixel-wise segmentation accuracy of the boosted classifier, learned with and without appearance-independent features, as a function of K, the numbers of textons. The graph of results is shown in Figure 3.20. As one would expect, the extra flexibility accorded by the additional appearance-independent features gave a small but significant improvement in results. Also of note is that there is a definite peak in performance as a function of the number of textons. With too few textons performance is very poor, but also with too many textons the boosting algorithm was seen to over-fit.

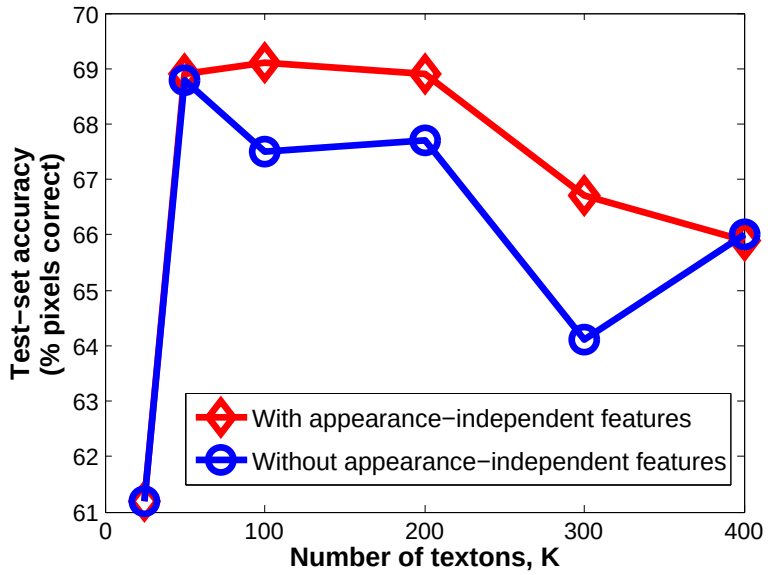

Figure 3.20: **Performance of appearance-independent features.** Performance against a test set in terms of pixel-wise segmentation accuracy is plotted against the number of textons, K. See text for explanation.

3.5.4 MSRC 21-Class Database Results

This section presents results of object class recognition and image segmentation for the full CRF model on the MSRC 21-class database. Our unoptimized implementation takes approximately three minutes to segment each test image. The majority of this time is spent evaluating all the shape-texture potentials $P(c|\mathbf{x}, i)$, involving a few thousand weak-classifier evaluations. Sub-sampling at test time by evaluating the shape-texture potentials on a $\Delta_{\text{ss}} \times \Delta_{\text{ss}}$ grid (with $\Delta_{\text{ss}} = 5$) produces results almost as good in about twenty-five seconds per test image.

Training the model took about 42 hours for 5000 rounds of boosting on the 21-class training set of 276 images on a 2.1 GHz machine with 2GB memory.[5] Without random feature selection, the training time would have been around 14000 hours. Note that due to memory constraints on this large dataset, the training integral images had to be computed on-the-fly which slowed the learning down by at least a factor two.

Example results of simultaneous recognition and segmentation are shown in Figures 3.1 and 3.21. These show both the original photographs and the color-coded output labeling.

[5]Simple optimizations subsequent to these experiments have reduced test time to about 10 seconds per image for the shape-texture potentials, and 20 seconds per image for the CRF inference. Training time was also reduced drastically to about 4 hours.

Note the quality of both recognition and segmentation. For instance, despite large occlusions, bicycles are recognized and segmented correctly, and large variations in the appearance of grass and road are correctly modeled.

In order to better understand the behavior of our algorithm, we also present some examples which work less well. In Figure 3.22(a) a very dark dog has been incorrectly classified as a cow. However, note that the segmentation accuracy is still very high; similarly Figure 3.22(c) gets a good segmentation, due to the strong color model in the CRF, but fails to get the semantic labels correct. In Figure 3.22(b) a large wooden boat was incorrectly classified as tree. Once again the segmentation mask is not bad. In Figure 3.22(d) the dog's shadow has been classified as building. This shows that the proper modeling of shadow is required. Finally, Figure 3.22(e) shows how the algorithm struggles with a dark and unusual image of water, due to the fact that water of this appearance does not occur in the training set.

Quantitative Evaluation

Figure 3.23 shows the confusion matrix obtained by applying our algorithm to the test set. Accuracy values in the table are computed as the percentage of image pixels assigned to the correct class label, ignoring pixels labeled as void in the ground truth. The overall classification accuracy is 72.2%; random chance would give $1/21 = 4.76\%$, and thus our results are about 15 times better than chance. For comparison, the boosted classifier alone gives an overall accuracy of 69.6%, thus the color, edge and location potentials increase the accuracy by 2.6%. This seemingly small numerical improvement corresponds to a large perceptual improvement (cf. Figure 3.19). The parameter settings, learned against the validation set, were $M = 5000$ rounds, $K = 400$ textons, $N_{\mathrm{R}} = 100$ candidate shapes, edge potential parameters $\boldsymbol{\theta}_\phi = [45, 10]^T$, filter-bank scale $\kappa = 1.0$, and location potential power $w_\lambda = 0.1$.

The greatest accuracies are for classes which have low visual variability and many training examples (such as grass, book, tree, road, sky and bicycle) whilst the lowest accuracies are for classes with high visual variability and fewer training examples (for example boat, chair, bird, dog). We expect more training data to boost considerably the recognition accuracy for those difficult classes. Additionally, using features with better lighting invariance properties should help considerably.

Let us now focus on some of the largest mistakes in the confusion matrix, to gather some intuition on how the algorithm may be improved. Structured objects such as airplanes,

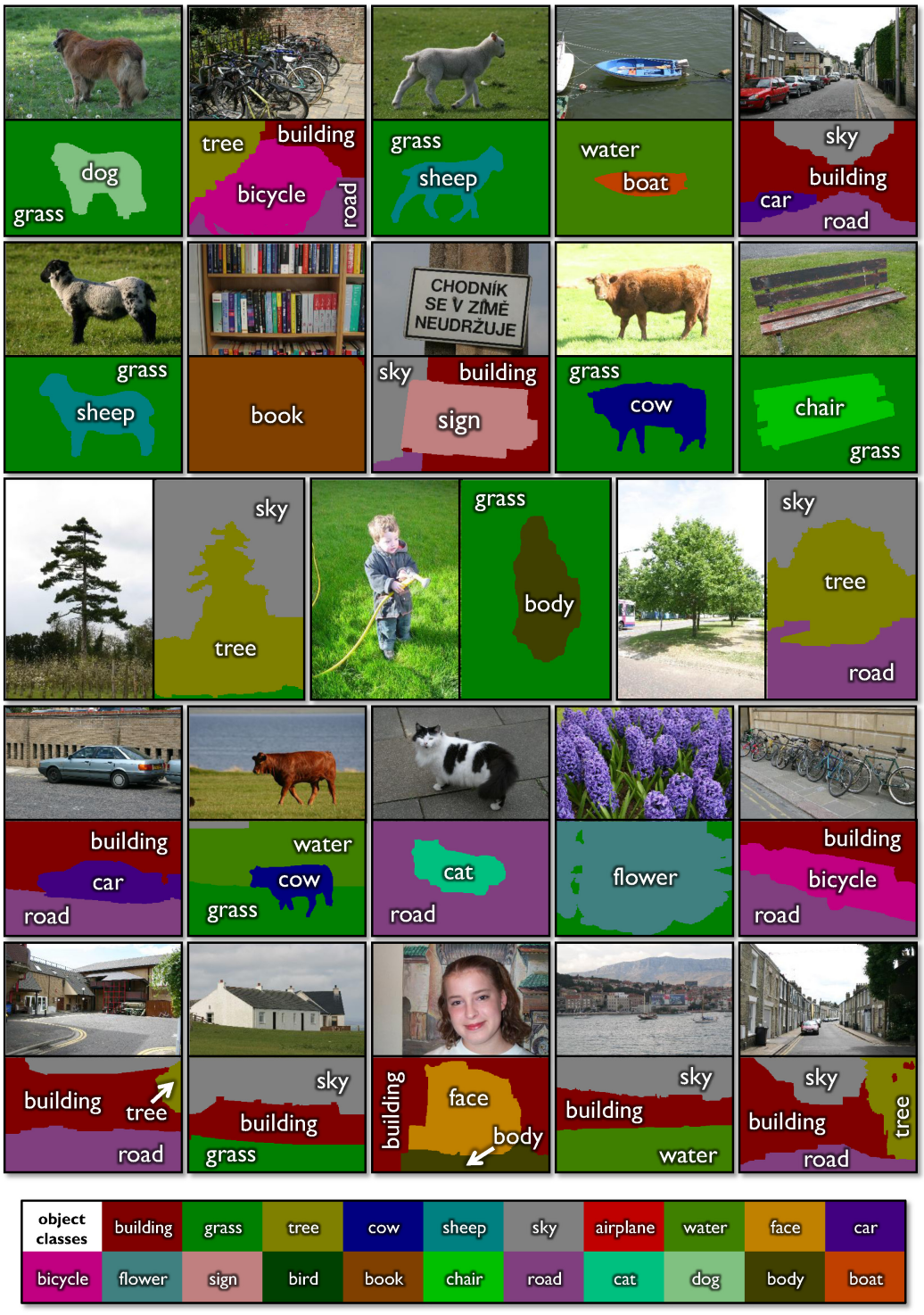

Figure 3.21: **Example results on the MSRC 21-class database.** Above, test images with inferred color-coded output object-class maps. Below, color-coding legend for the 21 object classes. For clarity, textual labels have also been superimposed on the resulting segmentations. Note that all images were processed using the *same* learned model. Further results from this system are given in Figure 3.1.

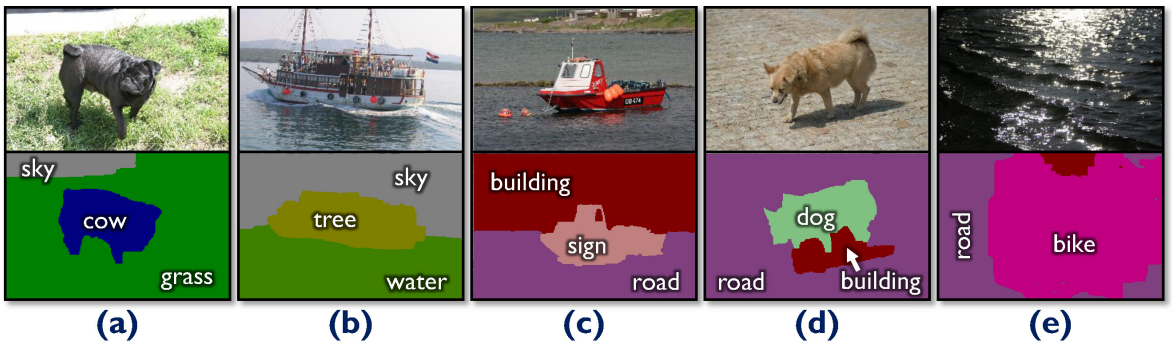

Figure 3.22: **Some examples where recognition works less well.** Input test images with corresponding color-coded output object-class maps. Note that even when recognition fails segmentation may still be quite accurate.

chairs, signs, boats are sometimes incorrectly classified as buildings. This kind of problem may be ameliorated using a parts-based modeling approach, such as [Winn & Shotton, 2006]. For example, detecting windows and roofs should resolve many such ambiguities. Furthermore, objects such as cows, sheep and chairs, which in training are always seen standing on grass, can be confused with grass. This latter effect is partially attributable to inaccuracies in the manual ground truth labeling, where pixels belonging to such classes are often mislabeled as grass near the boundary.

Separable TextonBoost

We investigated performance on the MSRC 21-class database using the separable 1D TextonBoost described towards the end of Section 3.4.2. Since this uses only the shape-texture potentials, we compare it with the 2D shape-texture potentials only result. For the full, joint model, recall that we obtained 69.6% pixel-wise segmentation accuracy. Using the separable model, we obtain the respectable 64.9%. Part of the success is due to the separable TextonBoost being good at getting the larger regions of classes such as sky and grass. Using this 1D method, there is a very noticeable speed-up of over 5 times for both training and test time. With optimizations, this speed improvement could be increased dramatically since the critical path of the algorithm has reduced from $O(NM)$ to $O(N + M)$ for an $N \times M$ image. Separable TextonBoost also requires considerably less memory during training, and so more training data could be employed if available.

True class \ Inferred class	building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bike	flower	sign	bird	book	chair	road	cat	dog	body	boat
building	**61.6**	4.7	9.7	0.3		2.5	0.6	1.3	2.0	2.6	2.1		0.6	0.2	4.8		6.3	0.4		0.5	
grass	0.3	**97.6**	0.5								0.1									1.3	
tree	1.2	4.4	**86.3**	0.5		2.9	1.4	1.9	0.8	0.1							0.1		0.2	0.1	
cow		30.9	0.7	**58.3**				0.9	0.4			0.4			4.2					4.1	
sheep	16.5	25.5	4.8	1.9	**50.4**									0.6			0.2				
sky	3.4	0.2	1.1			**82.6**		7.5									5.2				
aeroplane	21.5	7.2				3.0	**59.6**	8.5													
water	8.7	7.5	1.5	0.2		4.5		**52.9**		0.7	4.9			0.2	4.2		14.1	0.4			
face	4.1		1.1						**73.5**	7.1					8.4			0.4	0.2	5.2	
car	10.1		1.7							**62.5**	3.8		5.9	0.2			15.7				
bike	9.3		1.3							1.0	**74.5**		2.5			3.9	5.9		1.6		
flower		6.6	19.3	3.0								**62.8**			7.3		1.0				
sign	31.5	0.2	11.5	2.1		0.5		6.0		1.5		2.5	**35.1**		3.6	2.7	0.8	0.3		1.8	
bird	16.9	18.4	9.8	6.3	8.9	1.8		9.4						**19.4**			4.6	4.5			
book	2.6		0.6						0.4			2.0			**91.9**					2.4	
chair	20.6	24.8	9.6	18.2		0.2					3.7				1.9	**15.4**	4.5		1.1		
road	5.0	1.1	0.7					3.4	0.3	0.7	0.6		0.1	0.1	1.1		**86.0**			0.7	
cat	5.0		1.1	8.9				0.2		2.0					0.6		28.4	**53.6**	0.2		
dog	29.0	2.2	12.9	7.1				9.7							8.1		11.7		**19.2**		
body	4.6	2.8	2.0	2.1	1.3	0.2			6.0	1.1					9.9		1.7	4.0	2.1	**62.1**	
boat	25.1		11.5			3.8		30.6		2.0	8.6		6.4	5.1			0.3				**6.6**

Figure 3.23: **Accuracy of segmentation for the 21-class database.** Confusion matrix with percentages row-normalized. The overall pixel-wise accuracy is 72.2%.

Comparison with Winn *et al.*

To assess how much the shape and context modeling help with recognition we have compared the accuracy of our system against the framework of [Winn *et al.*, 2005], i.e. given a (manually) selected region, assign one single class label to it and then measure classification accuracy. On the 21-class database, our algorithm achieves 70.5% region-based recognition accuracy beating [Winn *et al.*, 2005] which achieves 67.6% using 5000 textons and their Gaussian class models. Moreover, the significant advantages of our proposed algorithm are that: (i) no regions need to be specified manually, and (ii) a pixel-wise semantic segmentation of the image is obtained.

3.5.5 Comparison with He *et al.*

We have also compared our results with those of [He *et al.*, 2004] on their Corel and Sowerby databases, as shown in Table 3.1 and Figure 3.24. For both models we show the results of the unary classifier alone as well as results for the full model. For the Sowerby database the parameters were set as $M = 6500$, $K = 250$, $\kappa = 0.7$, $\boldsymbol{\theta}_\phi = [10, 2]^T$, and $w_\lambda = 2$. For the Corel database, all images were first automatically color and intensity normalized, and the training set was augmented by applying random affine intensity changes to give

	Accuracy		Speed (Train/Test)	
	Sowerby	Corel	Sowerby	Corel
TextonBoost – full CRF model	88.6%	74.6%	20 m / 1.1 s	30 m / 2.5 s
TextonBoost – shape-texture only	85.6%	68.4%		
He *et al.* – mCRF model	89.5%	80.0%	24 h / 30 s	24 h / 30 s
He *et al.* – unary classifier only	82.4%	66.9%		

Table 3.1: **Comparison of segmentation/recognition accuracy and efficiency.** Timings for [He *et al.*, 2004] are from correspondence with authors.

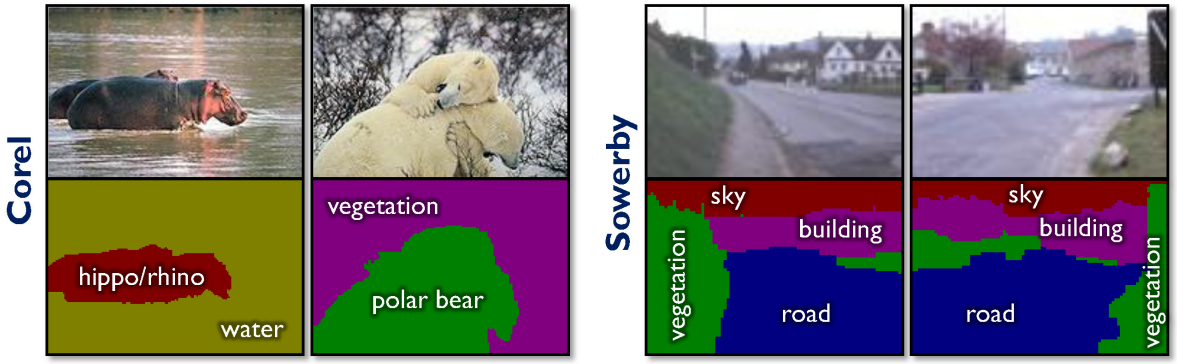

Figure 3.24: **Example results on the Corel and Sowerby databases.** A different set of object class labels and thus different color-coding is used here. Textual labels are superimposed for clarity.

the classifier improved invariance to illumination. The parameters were set as $M = 5000$, $K = 400$, $\kappa = 0.7$, $\boldsymbol{\theta}_\phi = [20, 2]^T$, and $w_\lambda = 4$.

Our method gives comparable or better (with only unary classification) results than [He *et al.*, 2004]. However, the careful choice of efficient features and learning techniques, and the avoidance of inefficient Gibbs sampling, enables our algorithm to scale much better with the number of training images and object classes. Incorporating *semantic* context information as in [He *et al.*, 2004] is likely to further improve our performance.

In the Corel database, the ground truth labeling between the ground and vegetation classes was often quite ambiguous to human observers. The confusion matrix of our results also bore this out, and merging these two classes results in significantly improved performance: 75.9% with just the shape-texture potentials, and 82.5% with the full CRF model.

3.5.6 Japanese Television Sequences

A separate model was trained for each of the nine Japanese television video sequences. A model could have been trained across all sequences simultaneously, but for the application of semantically segmenting a known television series, the extra flexibility of a generic

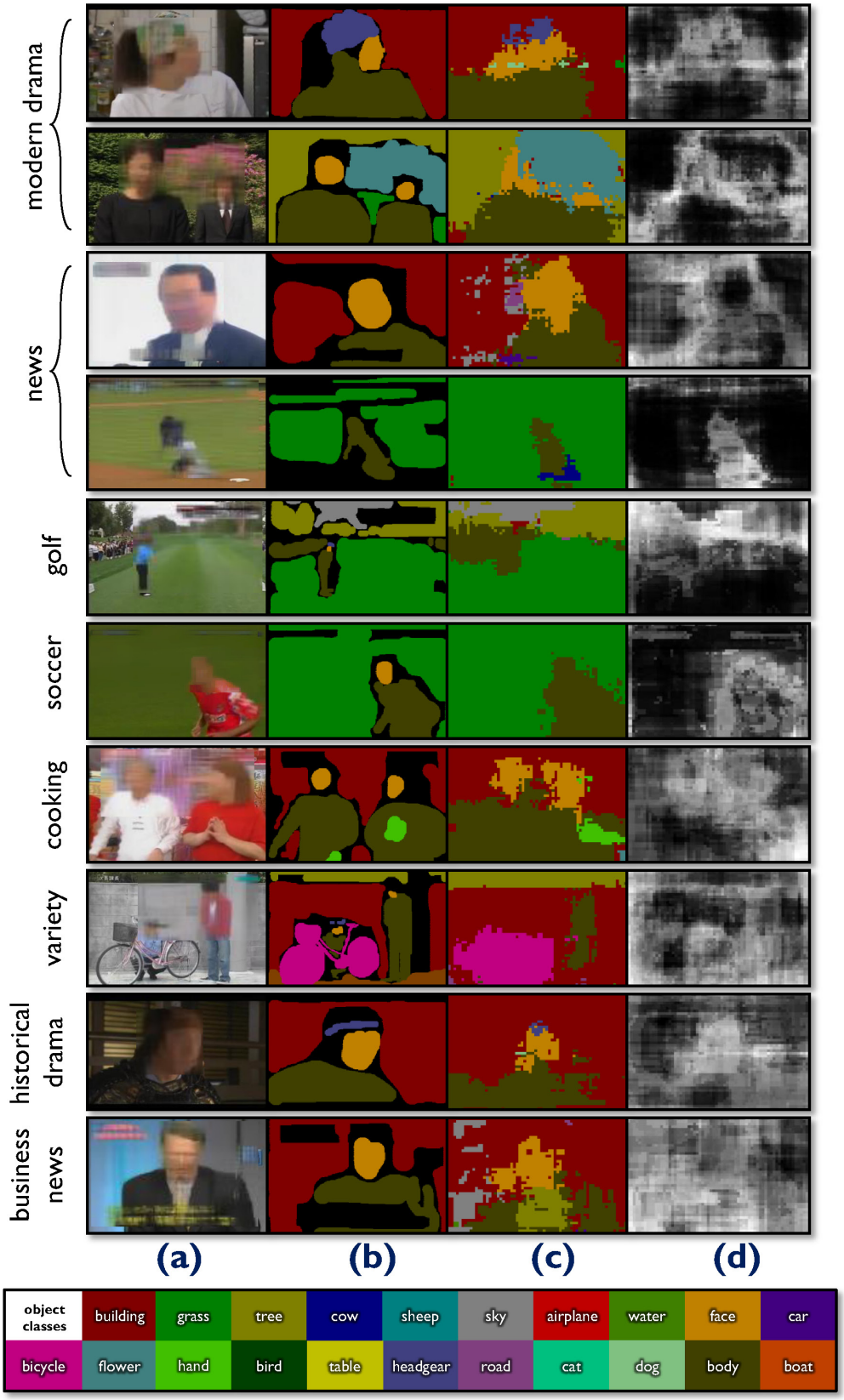

Figure 3.25: **Example results on the Japanese television sequences. (a)** The test image. Note that faces and text have been blurred out in this figure for copyright reasons. **(b)** The hand-labeled ground truth. **(c)** The most likely labels inferred by the shape-texture potentials. **(d)** The entropy of the inferred class label distributions; white is high entropy, black low entropy.

Sequence	Accuracy
Modern Drama	67.8%
News	67.8%
Golf	71.0%
Soccer	92.4%
Cooking	68.7%
Variety	64.2%
Music	60.5%
Historical drama	70.5%
Business news	58.1%

Table 3.2: **Quantitative results from evaluation on the Japanese television sequences.** Figures show pixel-wise segmentation accuracy.

model is unnecessary and would probably slightly worsen performance. Approximately 120 frames in each sequence were selected randomly. Half of these were used for training, and the other half used for testing. The training data was combined with the MSRC training data, and the shape-texture potentials were learned by boosting. Only the shape-texture potentials, as the most important part of the model, were used for evaluation. For more polished and visually pleasing results, the full CRF inference could be run, though as illustrated in Figure 3.19 only a small quantitative improvement would be seen. The parameters were set as $M = 700$, $K = 400$, and $\kappa = 0.7$.

Quantitative results of the overall segmentation accuracy are given in Table 3.2, and some qualitative results are given in Figure 3.25. The numbers show considerable accuracy across very varied sets of images, with on average two-thirds of all pixels being correctly classified into one of 21 classes, indicating significant potential for the application of TextonBoost to automatic analysis of video sequences. As can be seen from the images, the technique works well across very varied sequences. One slight limitation is that the system tends to get the larger objects in the scene classified correctly, but smaller objects such as hands can get missed off. This is at least partially due to the filter bank used during the textonization: the width of the Gaussian blur tends to over-smooth small objects.

The particularly strong result for the soccer sequence is perhaps slightly skewed by the large amount of grass present in the images. Additionally, due to the random selection of training and test images from the sequences, there are probably a few test frames that look extremely similar to training frames. This would skew the results slightly positively; perhaps further evaluation could employ on multiple episodes of the same television series.

Figure 3.26: **Semantic Photo Synthesis**. **Left:** user-drawn query. Here, the user has requested a photograph that has a large region of water underneath the Taj Mahal. **Right:** example automatic semantic photo synthesis results.

3.6 Applications

We briefly discussed general applications of visual recognition in Section 1.2. In this section we mention further exciting applications of TextonBoost and the concept of semantic segmentation.

AutoCollage

The work of [Rother *et al.*, 2006] takes a collection of images and automatically blends them together to create a visually pleasing collage; by choosing image regions of particular interest to humans (such as faces), detected through semantic segmentation, a more interesting collage could be generated. Additionally, images could be placed in suitable regions of the collage, so that for example, images with sky might be placed towards the top of the image.

Semantic Photo Synthesis

In [Johnson *et al.*, 2006], the user draws both particlar objects (the Taj Mahal, for example) and regions assigned a particular semantic label (sky, water, car, etc.) onto a canvas. The system then automatically queries a database containing images labeled by TextonBoost, to find relevant images that match the user-drawn query. Finally, it creates novel photo-realistic images by stitching together the image fragments that matched the individual parts of the query. Two example results of photo synthesis are shown in Figure 3.26.

Interactive Semantic Segmentation

An optimized implementation of our system could be used as a complete interactive semantic segmentation tool, as demonstrated in Figure 3.27. With only *one* user click on the incorrectly labeled part of the building, a correct and accurate segmentation was achieved.

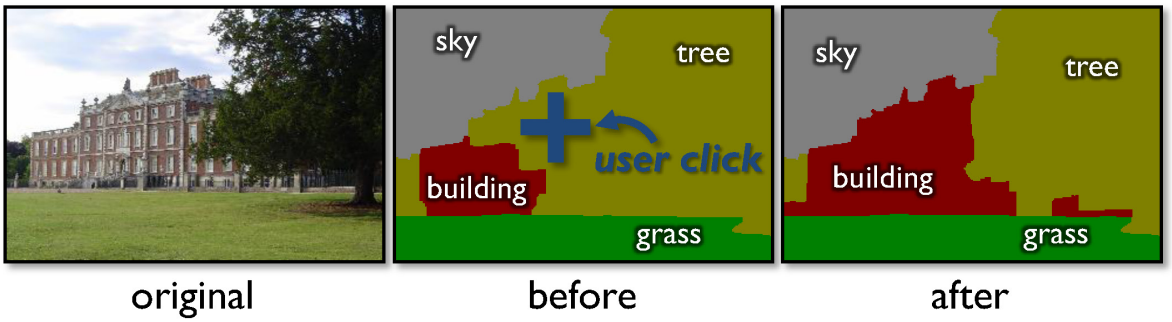

Figure 3.27: **Interactive object labeling**. **Left:** input test image. **Middle:** for this example, the automatic recognition failed to classify the building correctly. The user then clicks the mouse at the blue cross, stating that this part of the image is currently misclassified. **Right:** after this one click, the recognition algorithm is run again and now the building is now correctly classified and the segmentation is improved.

Internally, the unary potential of pixels within a small radius of the clicked pixel is set to infinity for its initial label, tree. The result of the graph cut optimization for this new CRF energy is the correct labeling. A further speed-up can potentially be achieved by re-using the flow of the previous solution, as described for the binary interactive systems in [Boykov & Jolly, 2001], or similarly to the method of [Kohli & Torr, 2005].

Interactive Image Editing

We suggest one final exciting application: interactive image editing. Imagine that TextonBoost produces a perfect semantic segmentation of an image. It is then possible to tailor image editing tools presented to the user according to the semantic type: for example, tools for editing 'sky' could allow the user to tint it more blue or increase the contrast; for foreground objects, such as the person in Figure 3.28, options could be given to automatically erase the object from the image (using image in-painting, for example [Criminisi *et al.*, 2004]), change the focus of background, fix red eye, or adjust the color balance just for that object.

3.7 Conclusions

This chapter has presented a novel discriminative model for efficient and effective recognition and simultaneous semantic segmentation of objects in images. We have: (i) introduced new features which simultaneously capture appearance, shape and context information, and shown that they outperform other existing techniques for this problem; (ii) trained our model efficiently by exploiting both randomized boosting and piecewise training techniques; and (iii) achieved efficient labeling by a combination of integral image processing

Figure 3.28: **Semantic-aware interactive image editing. Left:** the semantic segmentation given by TextonBoost is used to drive a semantic-aware user interface. Here the user has selected the person, and a context sensitive menu presents editing options specific for the 'person' class. **Right:** after the user clicks 'Erase...', an automatic system [Criminisi *et al.*, 2004] removes the person from the image by filling in over the top of her.

and feature sharing. The result is an algorithm which accurately recognizes and segments a large number of object classes in photographs much faster than existing systems. We have performed a thorough evaluation of the system on several varied image databases and have achieved accurate and competitive results.

To encourage and stimulate the application of and further development into the TextonBoost system, source code has been made available at [Shotton].

CHAPTER 4

CONTOUR AND TEXTURE COMBINED

4.1 Introduction

We have demonstrated in the preceding chapters the power of both contour and texture as recognition cues. In this chapter, we return to the tasks of classification and detection addressed in Chapter 2, and show that the combination of the contour-based features of Chapter 2 and the texture-based features of Chapter 3 can give superior recognition performance than either individually.

Chapter 2 demonstrated the power of contour fragments. The invariance properties of these features allow accurate recognition, even for classes of highly varied surface color and texture. However, contour fragments are not suited for detecting background. In contrast, color provides significant semantic cues. For example, a blue or green image region is very unlikely to represent a horse, even if the edges present resemble a horse. Additionally, contextual information, such as the fact that cars often appear above a road surface, cannot be exploited using contour-based methods, except perhaps where background edges repeatably co-occur with the object.

The texture-based features, called shape filters, presented in Chapter 3, gave excellent results for multi-class semantic segmentation. Shape filters characterize the color and texture of regions of image, and, as opposed to contour fragments, *are* particularly useful for detecting background and exploiting appearance context. Clearly, contour and texture features complement each other, and in this chapter we demonstrate their powerful synergy.

Given the groundwork of the previous two chapters, the combination of feature types proves particularly simple. An overview of the algorithm is given in Figure 4.1. Additionally, we demonstrate *cost-based learning*, whereby feature selection is steered both by the error on the training set and also the computational cost of the potential features at test time.

We show that cost-based learning can reduce run-time cost considerably while maintaining good quantitative performance.

We discuss in Section 4.2 the adaptation of shape filters for multi-scale recognition. Then in Section 4.3 we show how the learning and detection algorithms combine the responses of heterogeneous feature types in a principled manner. We compare performance of the individual and combined texture and contour features in Section 4.4, and conclude in Section 4.5.

Related Work

We briefly discuss work particularly related to the combination of feature types. Further references are given in Appendix A. The Normalized Cuts framework [Malik *et al.*, 2001] addressed bottom-up segmentation by combining contour features, based on orientation energy, and texture features, based on textons. Object recognition techniques have, for the most part, combined different interest point detectors and descriptor vectors, rather than fundamentally different feature types. Examples of such systems include [Zhang *et al.*, 2005a,b]. Some work has, however, successfully combined different types of features. In [Fergus *et al.*, 2004], a generative model of objects combines local SIFT descriptors [Lowe, 2004] with invariant curve descriptors. This performs well, although, as a constellation model [Fergus *et al.*, 2003], it scales badly with the number of parts. In [Kumar *et al.*, 2004], the outline contour and the interior texture were combined in a pictorial structures model, learned from video sequences. Most recently, [Opelt *et al.*, 2006b] has combined contour-based features with local descriptors to good effect.

4.2 Adapting Shape Filters for Recognition

In this section, we describe the simple adaptation of shape filters, originally used for semantic segmentation in [Shotton *et al.*, 2006] and detailed in Section 3.4.2, for multi-scale object detection. Three adaptations are made. Firstly, note that the task is no longer to infer the class label of a particular pixel, but instead, as in Chapter 2, the presence or absence of an object at a particular centroid hypothesis. Therefore, shape filters are evaluated relative to the centroid hypothesis $(\mathbf{x}, s)$. Secondly, we must scale the shapes (here, rectangles) to the hypothesis scale s. The scale-normalized rectangle $r = (\mathbf{r}_{\mathrm{tl}}, \mathbf{r}_{\mathrm{br}})$ is therefore scaled up to $sr = (s\mathbf{r}_{\mathrm{tl}}, s\mathbf{r}_{\mathrm{br}})$. Finally, to make responses at different scales comparable, we normalize by

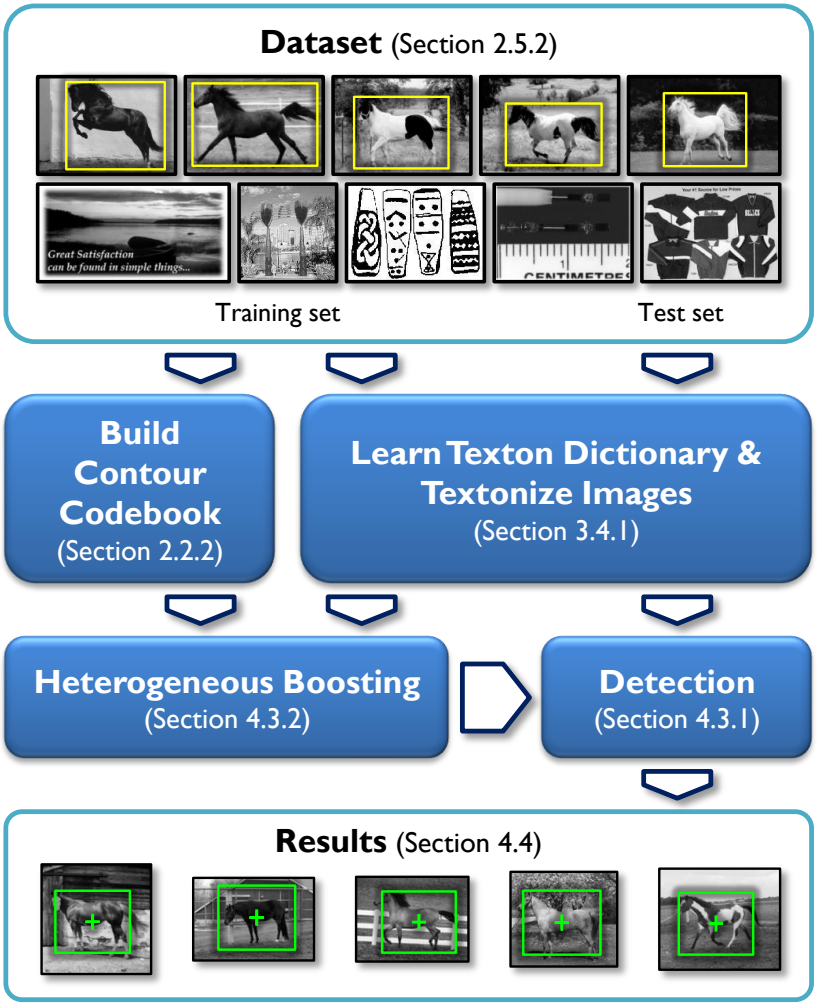

Figure 4.1: **Overview of the combined contour and texture detection algorithm.** A contour codebook is learned using the methods from Chapter 2. The images are textonized using the procedure from Chapter 3. A heterogeneous boosting algorithm learns how to combine contour and texture features into a discriminative classifier. Finally, the detection algorithm localizes objects in the test images. Compare with the detection algorithm outlined in Figure 2.2 and the segmentation algorithm outlined in Figure 3.3.

the area of the rectangle, so that the shape filter response becomes

$$v_{[r,t]}(\mathbf{x}, s) = \frac{1}{\text{area}(sr)} \sum_{j \in (sr+\mathbf{x})} [T_j = t] \tag{4.1}$$

where T_j represents the jth pixel of texton map T.

The new shape filters are illustrated in Figure 4.2, which demonstrates how these simple changes allow us to exploit shape, appearance, and appearance context for object detection. As we shall see in Section 4.4, these shape filters alone provide a strong recognition cue, but, in combination with contour fragments, result in considerably improved object recognition

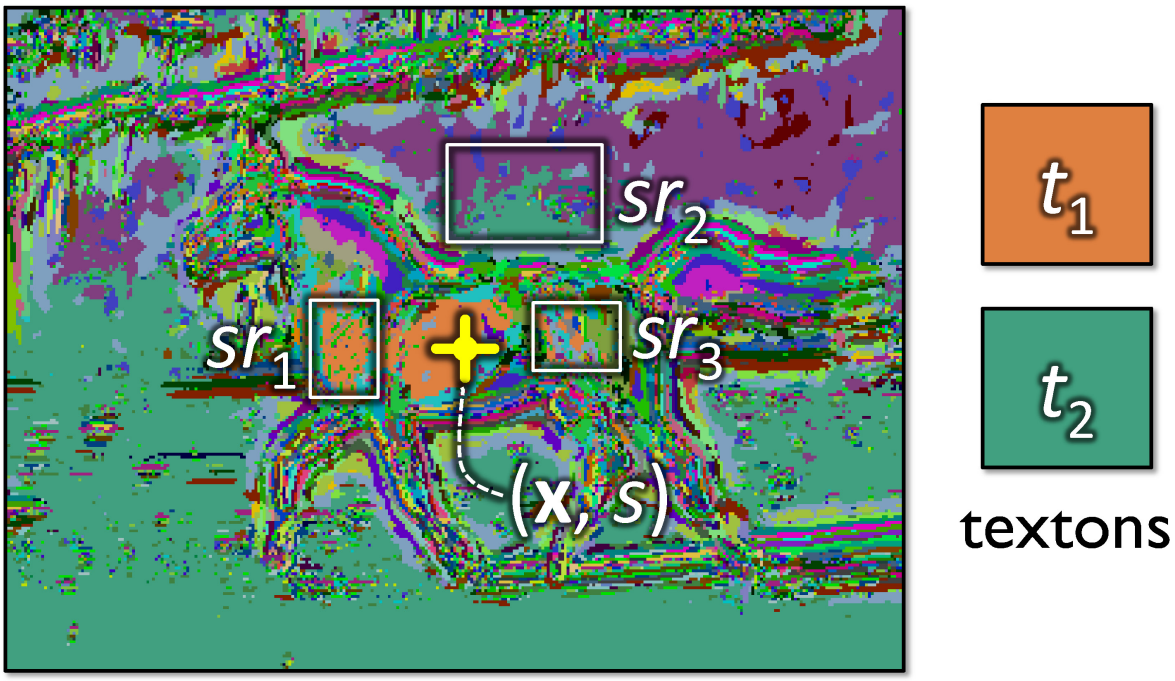

Figure 4.2: **Shape filters for recognition.** The texton map for a horse image is shown. Three example rectangles r_1, r_2 and r_3, each defined relative to the centroid hypothesis $(\mathbf{x}, s)$, are scaled by s. Shape filter (r_1, t_1) uses the object *appearance* by looking for the strong presence of texton t_1 (top right). Shape filter (r_2, t_2) exploits *appearance context*, since a strong response to texton t_2 (middle right) correlates with object presence at $(\mathbf{x}, s)$. The third shape filter, (r_3, t_2), exploits appearance in a more subtle way: the *absence* of texton t_2 (which we can identify as grass) in rectangle sr_3 gives evidence for the object being present at $(\mathbf{x}, s)$.

performance.

4.3 Heterogeneous Detection and Learning

In this section, we describe the modifications needed to extend detection and learning to use both contour and texture features.

4.3.1 Detection

The detection algorithm is identical to that in Section 2.3, other than a small change in the classification equation, which becomes (cf. (2.15)):

$$H(\mathbf{x}, s) = \sum_{m=1}^{M} h_m(\mathbf{x}, s) = \sum_{m=1}^{M} a_m[v_m(\mathbf{x}, s) > \theta_m] + b_m \ , \tag{4.2}$$

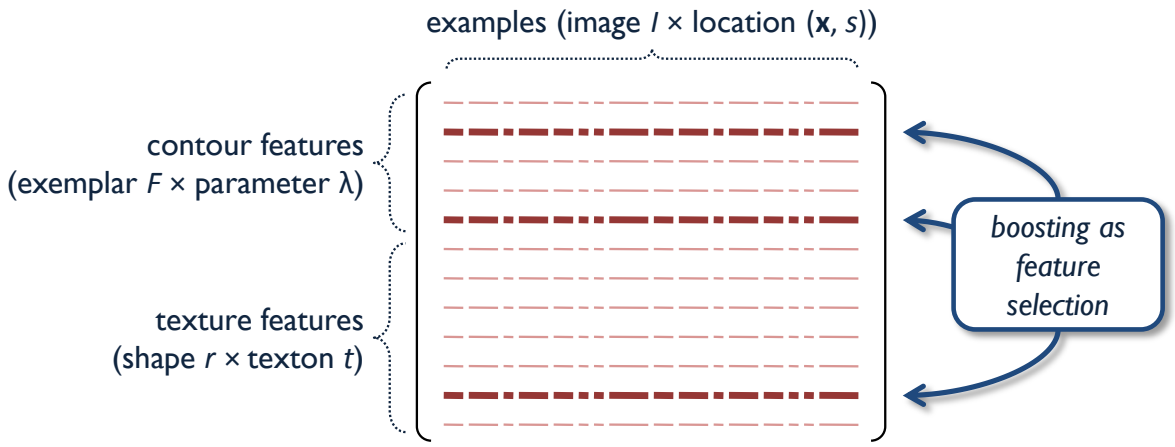

Figure 4.3: **Training matrix for boosting with combined features.** Each row contains responses for either **(upper)** a contour fragment, or **(lower)** a shape filter. Columns represent training examples. For illustration, each dash in the matrix corresponds to an image I, with length proportional to the number of example locations $\mathbf{x}$ in that image (see Figure 2.9). The boosting algorithm performs combined feature selection by iteratively choosing the row that minimizes a cost function (B.3). In this example, three rows (highlighted) were chosen: two contour features and one texture feature. Compare with Figure 3.14.

where the feature response v_m is:

$$v_m(\mathbf{x}, s) = \begin{cases} v_{[F_m,\lambda_m]}(\mathbf{x}, s) & \text{if round } m \text{ uses a contour feature} \\ v_{[r_m,t_m]}(\mathbf{x}, s) & \text{otherwise.} \end{cases} \tag{4.3}$$

4.3.2 Learning

We showed in the preceding two chapters how boosting is used for feature selection of contour fragments and shape filters. In this chapter, we use the single-class Gentle AdaBoost algorithm, as used in Section 2.4, and detailed in Appendix B. Recall that boosting learns from a set of training examples that here correspond to image locations. Each example consists of a target class label and a feature vector. Each dimension in the feature vector corresponds to a particular feature. The set of all feature vectors is denoted the training matrix.

The algorithm is extended for heterogeneous feature selection by simply expanding the training matrix, as illustrated in Figure 4.3. The matrix is divided into two portions: in the upper portion are the contour fragment responses $v_{[F,\lambda]}(\mathbf{x}, s)$ from (2.14), and in the lower portion are the scale-invariant shape filter responses $v_{[r,t]}(\mathbf{x}, s)$ (4.1). Recall that both feature responses are already normalized to the range $[0, 1]$. The weak learners in (4.2) are decision stumps of the form $a[v > \theta] + b$ where threshold θ is chosen from a discrete set $\Theta_c \subset [0, 1]$

for contour features, or from $\Theta_{\mathrm{t}} \subset [0, 1]$ for texture features. Boosting repeatedly selects the most discriminative weak learner that minimizes the error (B.4) across the training examples (the columns of the matrix). Each resulting weak learner corresponds directly to either a contour or a texture feature. This heterogeneous feature selection mechanism could easily be extended to additional feature types, by further extending the training matrix. An alternative is to learn separate classifiers, and combine them post-hoc [Opelt *et al.*, 2006b], although this is not investigated here.

The upper contour portion of the matrix contains $|\mathcal{F}| \times |\Lambda|$ rows (about 1000), with $\mathcal{F}$ the set of all contour exemplars, and Λ a discrete set for orientation specificity λ from (2.8). The lower texture portion contains $N_{\mathrm{R}} \times K$ rows (about 20000), with N_{R} the number of shapes, and K the number of textons. To save on storage requirements, and given the low cost of computing shape filter responses on-the-fly, only the contour feature responses are precomputed. Additionally, randomization (Section B.4.2) is employed on the texture features only.

Cost-Based Learning

As we shall see in Section 4.4, the two feature types carry different computational costs: contour fragment responses are about ten times more expensive to evaluate than shape filters responses (after all pre-processing). The standard learning algorithm greedily selects the optimal weak learner, based on training set classification performance alone. However, if run-time speed is important, we can bias the weak learner selection with a cost associated with the feature type.

Let us write these costs for contour and texture features as Q_{c} and Q_{t} respectively, although it is only the ratio of these costs that is important. The original boosting minimization of (B.3) is modified, so that, at round m, weak learner h_m is selected as:

$$h_m = \arg\max_{h \in \mathcal{H}} \frac{1}{Q_h}(J'_{\mathrm{wse}} - J_{\mathrm{wse}}[h]) \;, \tag{4.4}$$

where $Q_h \in \{Q_{\mathrm{c}}, Q_{\mathrm{t}}\}$ according to the feature type of candidate weak learner h, and J'_{wse} denotes the total training error at the previous round. This maximizes the improvement in classification accuracy on the training set per unit of computation cost. Therefore, boosting selects the weak learner that gives the greatest reduction in training error, divided by the run-time cost of the feature.

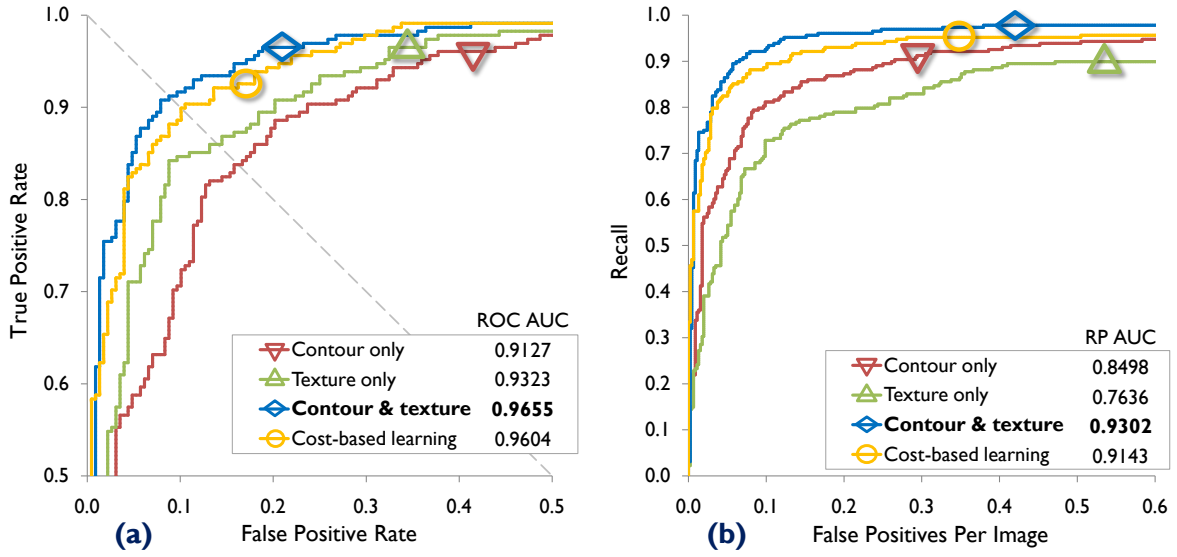

Figure 4.4: **Comparing contour and texture performance on the Weizmann horse test set. (a)** ROC curves showing classification performance. To aid readability only the top-left corner is shown. **(b)** RFPPI curves showing detection performance. Texture alone is better than contour at classification, but due to its poor localization, worse at detection. However, the combination of contour and texture is substantially better for both classification and detection. By weighting combined feature selection according to the computational cost of the features, cost-based learning enables performance almost as good as standard learning, but in a fraction of the cost.

In Section 4.4.1, we show that this procedure can increase run-time speed while maintaining good performance. A related cost-based learning approach based on decision trees and applied to real-time stereo is proposed in [Yin *et al.*, 2007].

4.4 Evaluation

We investigate the performance of the combined detector, using the same experimental procedure as that in Section 2.5.1. The same Weizmann and Graz 17 datasets from Section 2.5.2 are used, for comparison with the results of Section 2.5. These datasets are illustrated in Appendix C.

4.4.1 Multi-scale Weizmann Horses

For fair comparison, all parameters are kept the same as in Section 2.5.6. In particular, the number of features (homogeneous or heterogeneous) is fixed at $M = 100$.[1] The ad-

[1]This number of features is an order of magnitude smaller than used in Chapter 3. This is partly due to the simpler, binary, classification task, since even when sharing features, the number of weak learners required

ditional parameters needed to incorporate shape filters were set as follows: filter-bank scale factor $\kappa = 0.7$, $K = 200$ textons, and sub-sampling $\Delta_{\text{ss}} = 5$. The number of candidate rectangles was $N_{\text{R}} = 100$, and each scale-normalized rectangle r was randomly sampled from a uniform distribution to have $0.1 \leq r_{\text{w}} \leq 1.0$ and $0.1 \leq r_{\text{h}} \leq 1.0$ within rectangle $(-0.75, -0.75, 0.75, 0.75)$.

We show in Figure 4.4 the classification and detection results for the detector trained using (i) only contour, (ii) only texture, (iii) the combination of contour and texture, and (iv) the detector trained with cost-based learning. The combination of contour and texture features gives significantly superior performance to either individually, for both classification and detection. Interestingly, texture features are better individually at classification, while contour features are better individually at detection. This is probably because shape filters have fairly large spatial extent and are therefore poor at localizing the object. The combination seems to use contour fragments to accurately localize the object, and texture to further improve the classification confidence at that location.

We recorded the run-times in our unoptimized C# implementation: alone, contour took an average of 17.5 seconds per image, and texture took 4.6 seconds, while the combination took 12.5 seconds. In each of these timings, 3.4 seconds is spent executing the mean shift algorithm. The contour features are therefore approximately 12 times more expensive than the texture features. The combination is able to substantially increase quantitative performance above that achieved by contour features alone, while decreasing the computational cost. We further investigate run-time efficiency below, where we discuss the results for cost-based learning.

Feature Analysis

We examined the types of features chosen by the learning algorithm. Of the total 100 features selected, 65 were contour fragments and 35 were shape filters. This suggests that, for this dataset, contour is a slightly more useful recognition cue than texture, although both play an important role. We also computed the mean round number of each type of feature. These were 48 and 52 for contour and texture respectively, indicating a fairly even distribution of feature types in round number. Since more generic features are selected at earlier rounds of

for accurate classification increases with the number of classes. Additionally, the adapted shape filters (4.1) are explicitly scale-invariant, so that this invariance need not be learned implicitly at the cost of more features.

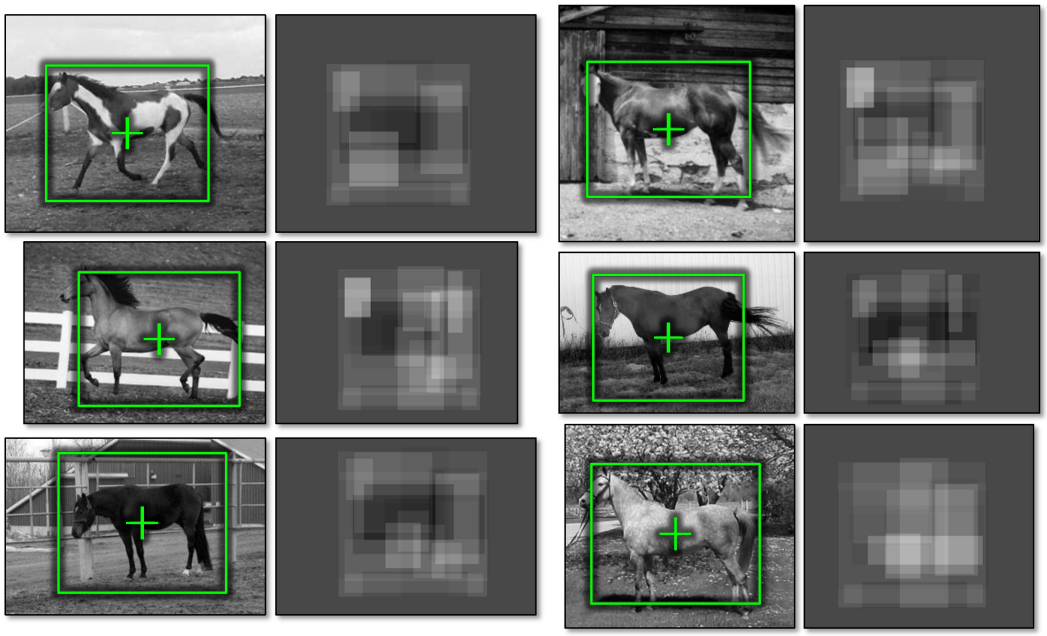

Figure 4.5: **Results and shape filter visualization for the Weizmann horse test set.** In the visualization, each shape filter (r, t) used is drawn using white to indicate the detected *presence* of texton t in image rectangle r, and black to indicate detected *absence* of t (see text). A general trend in these results is a black region over the body of the horse, although a counter-example is shown bottom-right. The black regions suggest that the large within-class textural variation prevents boosting from selecting features that use the presence of horse textures. Instead, it selects features that respond to the absence of certain textures, e.g. green grass or blue sky.

boosting, this result suggests similar generality of the contour and texture features.

Figure 4.5 shows a few example detections given by the combined detector, and visualizes the shape filters used. The visualization overlays each shape r, and indicates whether the presence (white) or the *absence* (black) of texton t positively contributed to the detection. This is determined by inspecting the weak learner $a[v_{[r,t]}(\mathbf{x}, s) > \theta] + b$. If classification confidence a is positive, it is the presence of texton t (quantified through response $v_{[r,t]}(\mathbf{x}, s)$ from (4.1)) that contributes positively to the classification, whereas if a is negative, it is the absence of t that contributes positively.[2] We see from the black regions in Figure 4.5 that the absence of particular textures on the horses' bodies contributes to their detections. This makes sense, given the extreme within-class textural variation of horses.

Figure 4.6 shows particular features selected by boosting. Of these features, rounds 1 & 3 are contour features corresponding to our notion of 'body', rounds 2 & 7 correspond

[2] In practice, the signs of a and b are always opposite, otherwise the weak learner would not improve the classification of the training set.

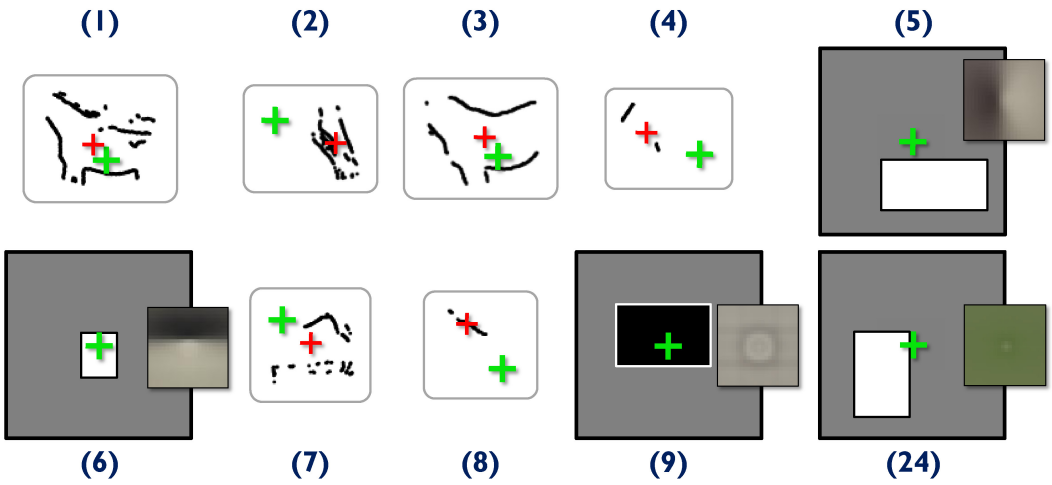

Figure 4.6: **Weak learners for the Weizmann horse dataset.** Ten weak learners of the combined detector are shown. The round numbers m are shown. In each, the green cross indicates the object centroid, and for contour fragments, the red cross indicates the fragment origin. For shape filters, we show inset right the texton visualisation (cf. Figure 3.9), and fill the rectangle white or black for texton presence or absence detection respectively (see text).

to 'hind legs', and rounds 4 & 8 to 'head and neck'. Of the texture features, round 5 looks for the presence of vertical edges in a region below the centroid ('legs'), round 6 looks for horizontal edges just below the centroid ('belly'), and round 9 looks for the absence of the the ring-like texton in a region surrounding the centroid ('body'). At round 24, the shape filter uses the presence of 'grass' as appearance context.

Cost-Based Learning

We show in Figure 4.4 the quantitative performance of the combined recognition system, with and without cost-based learning. As before, $M = 100$ cascaded weak learners are used. For this experiment, the relative costs were set as $Q_c = 5Q_t$. This resulted in 22 contour features and 78 texture features being chosen. We see that quantitative performance is slightly reduced, since fewer of the more discriminative, but expensive, contour features are chosen. However, the time per image is reduced to 7.8 seconds per image (from 12.5 seconds). Accounting for time taken performing the mean shift, this is a doubling in speed, with accuracy still substantially above what either feature type achieves individually. The average round number was 46 for the contour features, and 50 for the texture features, suggesting that the distribution of the feature types remains fairly even.

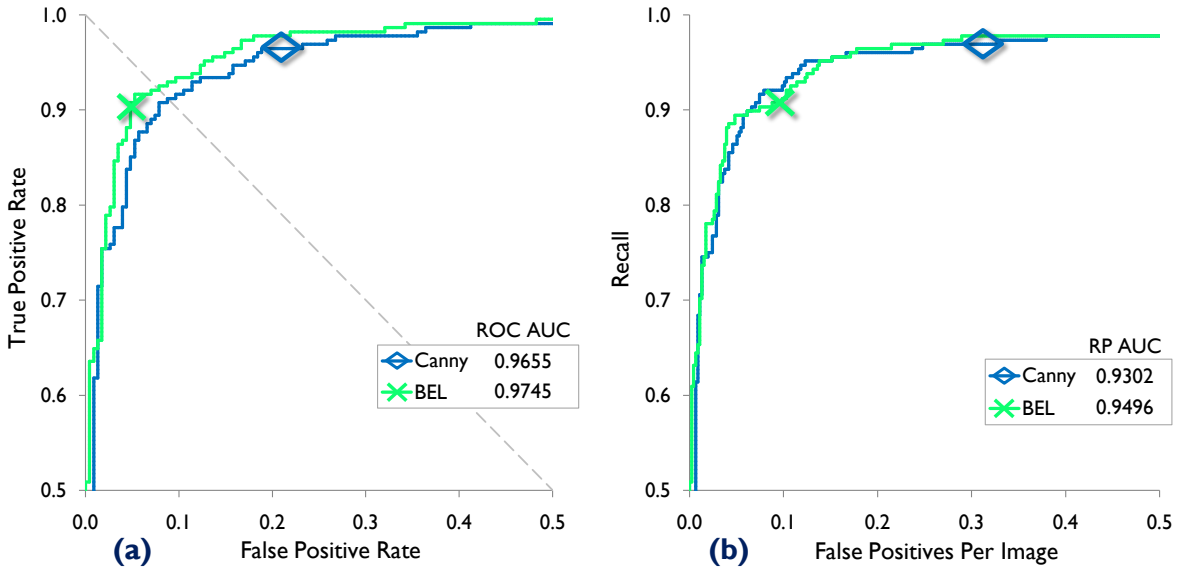

Figure 4.7: **Comparison between the edge detectors of Canny [Canny, 1986] and BEL [Dollár *et al.*, 2006].** Results are for the Weizmann horse dataset, using combined contour and texture features. **(a)** The ROC curves show that contour features derived from, and matched against, BEL edge maps noticeably improve classification performance in combination with texture features. **(b)** For detection, the RFPPI curves appear similar, although the RP AUCs show a slight improvement with BEL. A similar improvement in performance was observed in Section 2.5.8, using contour only.

Learned Edge Detection

The experiments so far have used the Canny edge detector [Canny, 1986]. The comparison in Section 2.5.8 showed that, for contour fragments, detecting edges using instead the boosted edge learning (BEL) of [Dollár *et al.*, 2006] considerably improved both classification and detection performance. We perform a similar experiment here, but use the combination of contour and texture features for recognition. The results in Figure 4.7 confirm that using the modern BEL technique noticeably improves performance, both for classification and detection, and gives the overall best results we obtain on the Weizmann horse dataset: 0.9745 ROC AUC for classification, and 0.9496 ROC AUC for detection. For the BEL result, 66 contour features and 34 texture features were selected.

4.4.2 Graz 17

Figure 4.8 shows classification and detection results for the Graz 17 dataset. The Canny edge detector is used throughout. Several trends are evident:

Classification: Classification performance is shown in Figure 4.8 (left). For roughly half

the classes, contour features alone give superior performance, while texture features alone give superior performance for the remainder. We see that, for almost every class, performance of the combined detector is at least as good as, and in some cases significantly better than, the performance of contour or texture alone. For several very challenging classes we achieve perfect or near perfect performance: airplanes, cars (rear), motorbikes, faces, cows (side), cows (front), and cups.

Detection: Detection performance is shown in Figure 4.8 (middle), and quantified as RP EER for comparison with [Opelt *et al.*, 2006c]. We reiterate the observation of Chapter 2 that detection is considerably harder than classification, since a precise localization must be achieved. The algorithm performs as well as, or significantly better than, [Opelt *et al.*, 2006c], for more than half of the classes: airplanes, cars (rear), motorbikes, faces, bikes (side), bikes (front), cars (front), bottles, cows, horses (side), cows(front), and cups. For two classes, motorbikes and cows (side), (near) perfect performance is achieved. For 14 of the classes, contour features alone perform better than texture alone, probably because shape filters are poor at precisely localizing objects. For the large majority of classes, the combination of contour and texture is better than either alone. There are still some classes for which performance is poor: bikes (rear), bikes (front), and people. We believe this is due to the very small numbers of training images for these classes (see Table 2.1). There does appear to be a general correlation between the number of training images and performance, and we postulate that results for classes with few training images would be improved considerably with more examples.

Feature Types: Shown in Figure 4.8 (right) are the proportions of contour and texture features used in the combined detector. These proportions are roughly equal for most classes, although certain classes show a significant bias. For motorbikes, faces, mugs, and cups, contour features are selected significantly more frequently. These classes do tend to have very distinctive contours, but less distinctive textures. Conversely, for cars (rear) and bikes (rear), more texture features are selected.

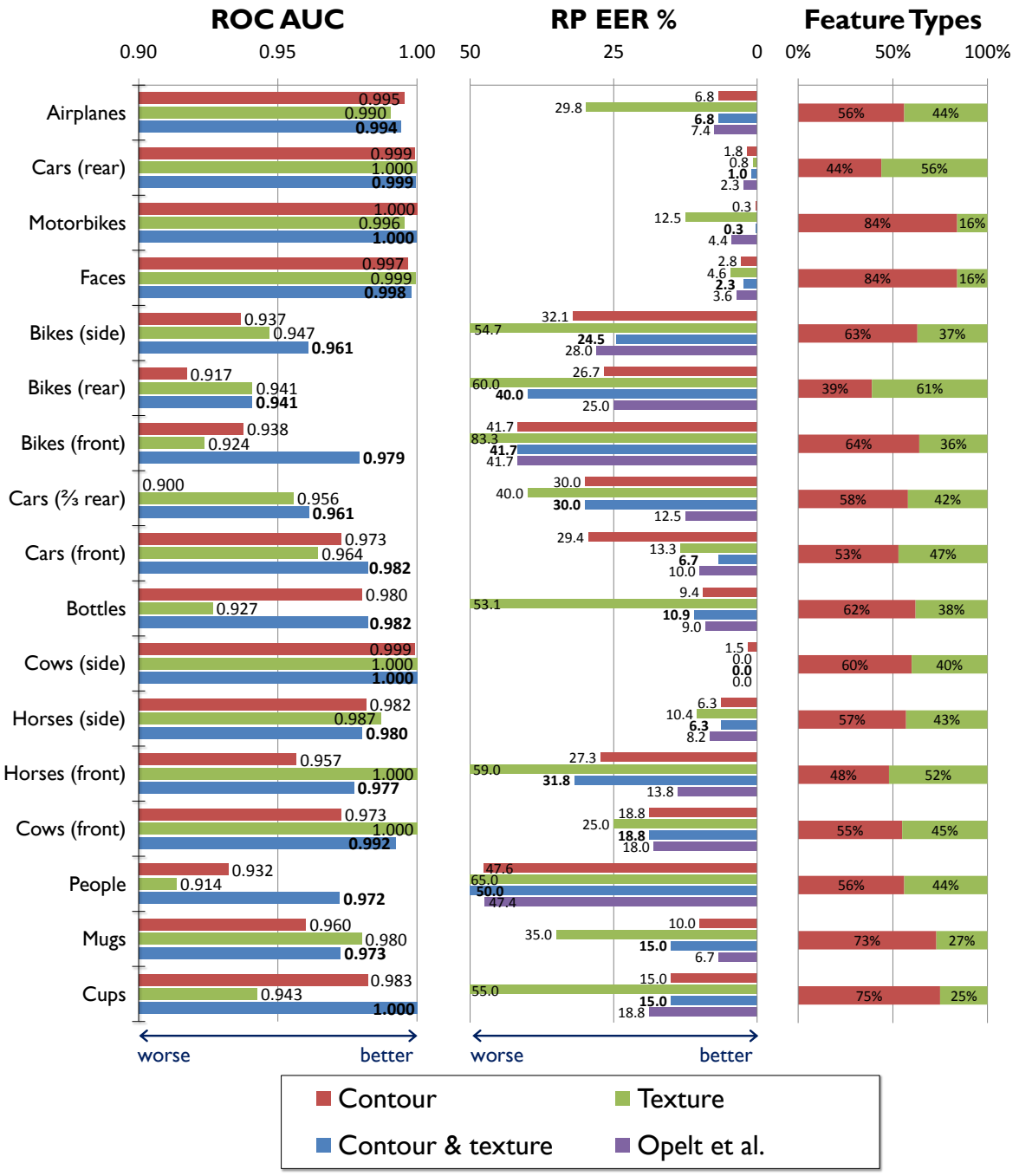

Figure 4.8: **Results on the Graz 17 dataset.** Performance is compared between only contour, only texture, the combination of contour and texture, and [Opelt *et al.*, 2006c]. **Left:** classification performance. **Middle:** detection performance. **Right:** the proportions of contour and texture features used in the combination. See text for analysis.

4.5 Conclusions

We have extended shape filters, proposed in Chapter 3, for multi-scale object detection. The thorough demonstration, on 17 challenging object classes, confirms our hypothesis that the boosted combination of shape filters with contour fragments can markedly improve results, above what either feature type can attain individually. Additionally, the combination considerably increases the speed of the detector, compared to using contour fragments alone. We saw how the detector can exploit both the presence and the absence of particular textures, and also how appearance context is harnessed. A new, cost-based learning algorithm was proposed, and shown to maintain good performance while significant reducing the computational cost.

CHAPTER 5

DISCUSSION

5.1 Findings

In Chapter 2 we proved experimentally that contour can be used successfully to perform image classification and categorical object detection. We demonstrated that our new approximate oriented chamfer distance gave superior performance to existing contour matching methods. This distance was used to learn a class-specific codebook of local contour fragments. We showed that boot-strapping by retraining on both the training and test data could improve performance and generalization. Additionally, the use of a modern learned edge detector substantially improved results over those attained using the Canny edge detector.

Chapter 3 investigated texture. We presented a new discriminative model based on a conditional random field, which was applied to the problem of semantic segmentation. We introduced a new set of features, shape filters, that can simultaneously represent appearance, shape, and context. We concentrated on the efficiency of training, exploiting randomized boosting, piecewise training, and sub-sampling. By using integral images and feature sharing, efficiency at test time was also achieved. We compared against other work, achieving competitive and visually pleasing results.

The cues of contour and texture were combined in Chapter 4, and applied to the tasks of classification and detection. Shape filters were extended for multi-scale detection. We showed that the combination of contour fragments and the extended shape filters significantly improved performance for most classes investigated, compared to either feature type individually. The combined detector was able to improve efficiency noticeably when compared to using contour fragments alone. By analyzing the features chosen by the heterogeneous boosting algorithm, we gained insight into how texture was being used: the presence

of particular textures, either on the object (utilizing appearance and shape) or on the background (utilizing appearance context), and the absence of particular textures, were both important. We proposed a form of cost-based learning, whereby the run-time cost of the features is used to bias feature selection. This maintained excellent performance while significantly increasing speed.

5.2 Limitations

While the techniques presented in this thesis have proved powerful, they are not able to cope with all the appearance variations that natural images present. In particular, the proposed classification and detection algorithms only tackle objects viewed from one angle, and the range of scales that can be handled is somewhat limited, both by computational efficiency and by the representational power of the features. The requirements for labeled data, bounding boxes for detection, and ground truth for semantic segmentation, are burdensome. The current implementations of the work presented here are also too slow for real-time applications.

The semantic segmentation method of Chapter 3 does not model a background class. It implicitly assumes that every pixel in a test image can be meaningfully assigned to one of the learned classes. This is an over-simplification of real-world images, since clearly, short of modeling thousands of classes, some regions of test images are likely not to correspond to any learned class.

We are still some way off a system that can cope adequately with the extreme examples of Figures 1.3 and 1.4. Substantial improvement in object representation and contextual modeling are required. As illustrated in Figure 3.22(b), we have started to reach the limits of the simple ontological model used in our work on semantic segmentation. That a single semantic class label is sufficient to completely describe a pixel is overly simplistic: should a dog pixel be labeled dog, pet, mammal, or all of the above? As the field moves towards more complicated datasets, improvements to ontologies will have more profound effects on performance.

5.3 Future Work

To address these limitations, we propose several avenues for future research.

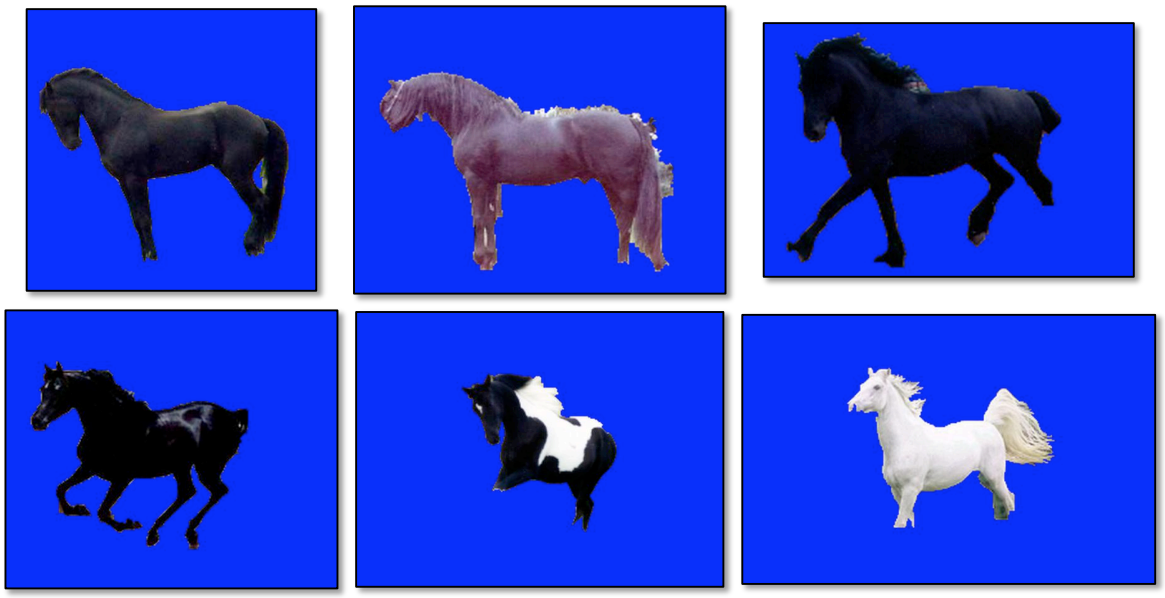

Figure 5.1: **Example GrabCut [Rother *et al.*, 2004] segmentations.** Our detection algorithm localizes the objects in scale-space with a bounding rectangle. This rectangle is used to initialize GrabCut, which automatically learns foreground and background color models and uses iterated graph cuts to segment the object from the background.

Contour: We are interested in developing a more probabilistic fusion of the classification probabilities from multiple sliding windows, as mentioned in Section 2.3.2. We plan to investigate further our codebook of contour fragments. The clustering algorithm used presently is inefficient, and perhaps an agglomerative clustering method would be faster. We are currently investigating how the codebook might be used in a bag-of-words recognition model, in place of sparse local descriptors. Our investigation of modern, learned edge detection algorithms is preliminary and more work is desirable there. Considerable optimization is possible. Given the results that show our approximate chamfer matching does not harm performance, perhaps down-sampled distance transforms could also be used.

A few preliminary results of combining our detection technique with a segmentation algorithm are presented in Figure 5.1. These results took the inferred object bounding rectangles as initializations to GrabCut [Rother *et al.*, 2004; Blake *et al.*, 2004]. We would like to further investigate this to see if individually segmented fragments could serve as a segmentation prior similar to [Kumar *et al.*, 2005]. An alternative method proposed in [Zheng *et al.*, 2007] is to learn to segment directly from the image.

Texture: We hope to integrate explicit *semantic* context information, in addition to the ap-

pearance context information used in Chapters 3 and 4. Work such as [He *et al.*, 2004] uses semantic context to improve segmentation accuracy. Our brief investigation into separable TextonBoost could be extended. Perhaps more interesting factorizations similar to (3.19) could be used, for example including diagonal terms. We would like to investigate other forms of textonization. For example, alternative filter banks including cross derivatives or clustered SIFT descriptors [Lowe, 2004] might provide extra representational flexibility and additional invariance properties. Also, other forms of clustering, such as [Jurie & Triggs, 2005], might improve results over k-means. A soft assignment of pixels to textons might produce better results. We are currently investigating simple extensions to shape filters to incorporate motion cues when applied to video sequences.

Combining Features: The proposed framework for heterogeneous learning and detection could be straightforwardly extended to incorporate additional cues, and of particular interest is the further fusion with sparse local descriptors. We also believe that an optimized real-time implementation is feasible. This would allow accurate tracking as detection; preliminary results in this direction are shown in Figure 5.2. Additionally, the detector could be trained to discriminate between different poses of the tracked object, for example, between open and closed hands.

5.4 Final Remarks

Visual recognition of object categories has advanced dramatically over the last few years. The community is gradually extending the range of variabilities, illustrated in Figures 1.3 and 1.4, that can be handled successfully. For example, new work into view-point invariant recognition is proposed in [Hoiem *et al.*, 2007]. The boundary between the appearance of objects at large scales and at small scales is of particular interest. As one zooms out from a single flower to a whole field of flowers, for example, at what point should one change from modeling the individual to modeling the group. These 'phase shifts' in appearance have not been adequately investigated, and perhaps a fusion of the detection and semantic segmentation methods presented in this work would help. The ability to cope with partially occluded objects is also of considerable concern, and some preliminary work in this direction is presented in [Winn & Shotton, 2006]. The forthcoming PASCAL Visual Object Challenge 2007 [VOC] will also try to push the envelope of achievable recognition tasks.

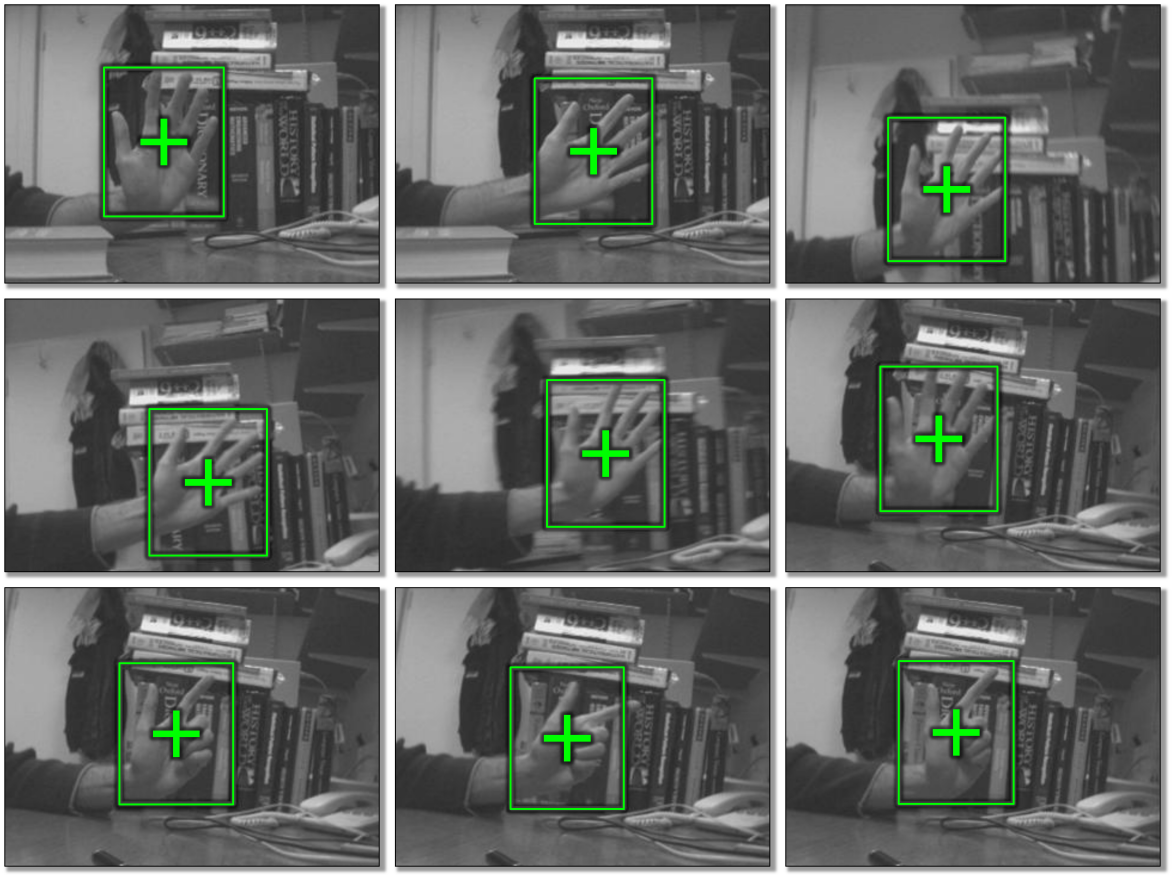

Figure 5.2: **Preliminary results of hand tracking as detection.** A real-time implementation of our combined detector would allow automatic hand tracking.

From a learning point of view, the challenges lie in exploiting unlabeled or partially labeled data. Such unsupervised and semi-supervised techniques will become more important as we move to more sophisticated ontological models that deal with more classes.

While we are still a considerable way from accurately recognizing the tens of thousands of classes that humans effortlessly distinguish despite incredible variations in appearance, we believe that this thesis has taken a positive step towards a solution.

APPENDIX A

BIBLIOGRAPHIC NOTES

This appendix briefly describes some of the important and exciting research related to this thesis. As far as possible, we have arranged this literature survey hierarchically, although not all papers are straightforwardly categorized. Within each section, work is presented roughly chronologically. When referencing a body of work presented over several papers, the most modern, definitive source is cited first.

This appendix is divided into four sections. Section A.1 discusses image features, including interest point detectors, local descriptors, contour-based features, and methods that combine feature types. In Section A.2 we discuss bottom-up and top-down segmentation algorithms, and also work on semantic segmentation. In Section A.3 we explore research into visual recognition, including image classification and categorical object detection. Finally, we discuss recent developments that combine recognition and segmentation, in Section A.4.

A.1 Image Features

This thesis has proposed two new types of image feature, contour fragments in Chapter 2, and shape filters in Chapters 3 and 4. In this section, we discuss alternative image features. In general, features are either computed densely, on a grid of points, or sparsely, at a set of *interest points*. Dense features can handle texture well (see e.g. Figure 3.2) and can be used for segmentation, but by their nature are computationally expensive. Sparse features are more efficient, since there are fewer of them per image. They also have invariance properties, such as affine geometric invariance, that have made them popular for solving the correspondence problem for wide-baseline matching, e.g. [Schaffalitzky & Zisserman, 2002a].

In Section A.1.1, we discuss methods that localize sparse features (i.e. detect interest points). Many techniques use *local descriptors* to describe image regions, and these methods are explored in Section A.1.2. We talk about contour-based features in Section A.1.3, and

texture-based features in Section A.1.4. The combination of features types is discussed in Section A.1.5.

A.1.1 Localizing Sparse Features

Sparse features, computed at interest points, allow a significant computational saving by focusing attention on salient image regions. All the algorithms discussed here use hand-designed low-level image cues, although recent work [Kienzle *et al.*, 2006] has suggested *learning* interest point detectors.

The Canny edge detector [Canny, 1986] can be viewed as a primitive form of interest point detector, and is still in active use today in, for example, Chapters 2 and 4 and [Ferrari *et al.*, 2006b; Mikolajczyk *et al.*, 2003]. However, most recent interest point detectors infer a region of spatial support at particular sparse points. Much effort has be focused on improving the Förstner or Harris-Stephens corner detector [Förstner & Gülch, 1987; Harris & Stephens, 1988] with various geometrical invariances. The Harris-Laplace detector of [Mikolajczyk & Schmid, 2001] added scale-invariance. Further research, relying on earlier work on shape-from-texture in [Lindeberg & Gårding, 1997], extended Harris-Laplace with affine invariance in [Mikolajczyk & Schmid, 2004, 2002; Baumberg, 2000]. Lowe used the difference of Gaussians operator to provide scale-invariant interest points for SIFT descriptors in [Lowe, 2004, 1999]. An affine invariant region detector was proposed in [Tuytelaars & Van Gool, 2000] that finds intensity extrema that are robust to large illumination changes. The scale saliency method from [Kadir *et al.*, 2004, 2003; Kadir & Brady, 2001] suggested localizing interest points at regions of high intensity entropy. Matas *et al*. proposed the very efficient and affine invariant Maximally Stable Extremal regions in [Matas *et al.*, 2002]. An extension of interest-points to spatio-temporal volumes was suggested in [Laptev & Lindeberg, 2003].

A.1.2 Local Descriptors

The development of affine invariant region detectors was matched by new local descriptors. These were designed to describe image regions in a fairly low-dimensional vector with certain geometric and photometric invariances. Local descriptors were used first for wide-baseline matching, and more recently for categorical recognition. We used one such local descriptor, SIFT (described shortly), for comparison against contour fragments, in Section 2.5.9. While descriptors are usually computed sparsely at interest points, they can be

computed densely, e.g. [Kapoor & Winn, 2006].

The simplest descriptors are raw patches of image, which are often matched using sum of squared differences, or normalized cross correlation. Patches were used for example in [Leibe & Schiele, 2003] for recognition, and in [Criminisi *et al.*, 2007] for dense-stereo. The image epitome model of [Jojic *et al.*, 2003; Cheung *et al.*, 2005] is a compact method for representing a dictionary of image patches. Recently, the Jigsaw model of [Kannan *et al.*, 2006] allows the compact representation of image patches with non-rectangular extents.

Beyond simple patches, many descriptors aim at invariance to rotation and affine intensity changes. Steerable Filters [Freeman, 1992] achieve rotation invariance by steering local image derivatives in the direction of the local image gradient. Moment invariants were proposed by [Van Gool *et al.*, 1996], and differential invariants by [Schmid & Mohr, 1997]. The SIFT descriptor was proposed in [Lowe, 2004, 1999]. This computes histograms of oriented gradients, and was shown to out-perform several other descriptors in the thorough comparison of [Mikolajczyk & Schmid, 2003]. Variants of SIFT were proposed in [Mikolajczyk *et al.*, 2005] and [Brown *et al.*, 2005]. The rotation invariant modulus of complex filters was used in [Schaffalitzky & Zisserman, 2002b]. A descriptor specifically designed to operate at sparse edge points for the recognition of tubular structures was proposed in [Mikolajczyk *et al.*, 2003].

A.1.3 Contour Features

Contour, defined in terms of the edges of an image, was already considered a useful cue for recognition in Marr's Primal Sketch [Marr, 1982]. The standard Canny edge detector [Canny, 1986] has recently been improved by learning edge detection from a set of training images in [Dollár *et al.*, 2006; Martin *et al.*, 2004].

Contour was first used for detecting particular objects, matched as complete, rigid templates. The Generalized Hough Transform [Ballard, 1981] is one matching method. Alternatively, the Chamfer [Barrow *et al.*, 1977] and Hausdorff [Huttenlocher & Rucklidge, 1992] distances have been used to detect and track articulated objects: people in [Gavrila, 2000; Felzenszwalb, 2001; Toyama & Blake, 2002], hands in [Stenger *et al.*, 2003; Thayananthan *et al.*, 2004], and the upper body in [Navaratnam *et al.*, 2005]. [Leibe *et al.*, 2005] used chamfer matched pedestrian outlines in a recognition verification stage. Techniques such as these require a large set of templates to represent all joint object configurations, and often a hierarchy is used for efficiency.

Alternative approaches use fragments of contour. Inspired by Cubist art, [Nelson & Selinger, 1998] used semi-invariant local keys to compare test images against a database. [Fergus *et al.*, 2004] augmented the constellation model (described below) with contour fragment features, but only exploited fairly clean, planar curves with at least two points of inflection. In [Kumar *et al.*, 2004], contour fragments learned from video sequences were arranged in Pictorial Structures and used for detection of articulated objects. Good results were obtained, although a fairly complex tracking of video sequences or manual labeling of parts was needed for learning. [Borenstein *et al.*, 2004] used both image and contour fragments for segmentation, though did not address classification or detection. A similar technique to [Shotton *et al.*, 2005] was subsequently presented in [Opelt *et al.*, 2006a].

Shape contexts [Belongie *et al.*, 2002] use a log-polar histogram of edgels for shape matching. A similar descriptor based on geometric blur was presented in [Berg & Malik, 2001], and was used for matching under considerable deformation in [Berg *et al.*, 2005]. A generative model of shape matching was proposed in [Tu & Yuille, 2004]. A classification cascade was used in [Carmichael & Hebert, 2004] to localize the edges of wiry objects. [Jurie & Schmid, 2004] proposed contour-based features that characterize the local edge convexity. In [Anderson *et al.*, 2006], rotation invariant object recognition was proposed that used clusters of wavelet-based edge features. The contour segment network [Ferrari *et al.*, 2006b,a] builds a graph connecting neighboring edge fragments.

A.1.4 Texture Features

Texture has long been seen as a useful image feature. Gabor filters were proposed in [Daugman, 1980], and have subsequently been used for iris recognition in [Daugman, 2003]. Malik & Perona presented a model of human preattentive texture perception in [Malik & Perona, 1990]. Textons, detailed in Chapter 3, were investigated in [Leung & Malik, 2001] and [Varma & Zisserman, 2005]. Kingsbury proposed a family of complex wavelets with shift invariant properties in [Kingsbury, 2001]. Leung proposed modeling the conditional distribution of database textons given image textons for recognition in [Leung, 2004]. Correlatons were used to model some rigid spatial relationships between textons in [Savarese *et al.*, 2006].

A.1.5 Combining Features

As investigated in Chapter 4, the combination of different feature types can dramatically improve performance. Video Google, a system for image-based search of videos, was pre-

sented in [Sivic & Zisserman, 2003] and combined different local descriptors. Similarly, different descriptors were combined in [Zhang *et al.*, 2005a,b]. Fergus *et al.* used both local descriptors and contour features for object detection in [Fergus *et al.*, 2004]. Pedestrian detection was tackled in [Leibe *et al.*, 2005], where a set of object detections based on image patches was post-processed using outline contours. Opelt *et al.* proposed augmenting their contour-based technique with local descriptors in [Opelt *et al.*, 2006b].

A.2 Segmentation

Chapter 3 addresses the task of semantic segmentation. We discuss in this section related work on segmentation and semantic segmentation. In Section A.2.1, we see that segmentation was originally posed as a bottom-up problem that used only low-level image information. Later, top-down, class-specific knowledge was incorporated (Section A.2.2), and then combined with bottom-up information (Section A.2.3). Most recently, semantic segmentation has become possible (Section A.2.4).

A.2.1 Bottom-Up

Normalized cuts [Shi & Malik, 1997; Malik *et al.*, 2001] incorporated both texture and edge based cues for bottom-up segmentation. Graph cuts were proposed for fast, accurate segmentation in [Boykov *et al.*, 2001; Kolmogorov & Zabih, 2004; Boykov & Jolly, 2001; Boykov *et al.*, 1999], and exploited for interactive image segmentation by the GrabCut system [Boykov & Jolly, 2001; Rother *et al.*, 2004; Blake *et al.*, 2004].

A.2.2 Top-Down

[Borenstein & Ullman, 2002] presented a novel class-specific segmentation algorithm based on matching image fragments. In [Ferrari *et al.*, 2004], initial local correspondences against a template initialized an 'image exploration', locating and segmenting the object even under deformation. Discriminative random fields were suggested by [Kumar & Hebert, 2003] and used for image de-noising and detecting man-made structure in images.[1] Weakly-supervised top-down segmentation was investigated in [Vasconcelos *et al.*, 2006].

[1] Discriminative random fields are simply conditional random fields [Lafferty *et al.*, 2001] over two-dimensional images.

A.2.3 Combined Top-Down & Bottom-Up

The combination of bottom-up and top-down segmentation cues was suggested by [Borenstein *et al.*, 2004; Borenstein & Ullman, 2004; Borenstein & Malik, 2006]. LOCUS [Winn & Jojic, 2005] presented an generative probabilistic model of segmentation, which unlike many other techniques does not require segmented training data. [Levin & Weiss, 2006] showed how to combine bottom-up and top-down cues in a conditional random field for segmentation.

A.2.4 Semantic Segmentation

[Konishi & Yuille, 2000] used color and texture statistics to achieve a semantic segmentation, with accurate but grainy results, since no spatial coherence was enforced. Ideas from machine translation were used in [Duygulu *et al.*, 2002] to get a rough semantic segmentation from training data labeled only with textual labels. Accurate but expensive data-driven Markov chain Monte Carlo was used in [Tu *et al.*, 2003] to give a coherent semantic scene analysis, and to specifically recognize text and faces. Conditional random fields were used for semantic segmentation by [He *et al.*, 2004, 2006]. A hierarchical field framework was proposed in [Kumar & Hebert, 2005]. Approximate three-dimensional geometric information was inferred from two-dimensional images (without using stereo) by exploiting geometric context, in [Hoiem *et al.*, 2005].

A.3 Recognition

Chapters 2 and 4 addressed two recognition sub-goals: image classification and categorical object detection. In this section, we discuss work that directly relates to these tasks. Most modern work recognizes objects as the sum of their constituent parts. We can notionally divide such parts-based methods into those that use spatial models (Section A.3.1), and those that do not, the so-called *bag-of-words* models (Section A.3.2). In Section A.3.3, we describe work that uses *context* to recognize objects.

A.3.1 Spatial Models

The constellation model of [Fergus *et al.*, 2003; Weber *et al.*, 2000; Burl *et al.*, 1998] learned a joint generative model of the layout of parts without requiring labeled parts. The statistics of wavelet-based parts were used in [Schneiderman & Kanade, 2004, 2000]. Class-specific

fragments of image patches were selected using mutual information in [Ullman *et al.*, 2001]. The pictorial structures model [Felzenszwalb & Huttenlocher, 2005] connected parts with virtual springs. Mixtures of trees were suggested by [Ioffe & Forsyth, 2001]. [Agarwal & Roth, 2002] suggested sparse image patches for recognition. The selection of scale-invariant parts using likelihood ratio and mutual information was investigated in [Dorkó & Schmid, 2005, 2003]. Humans were detected in [Mikolajczyk *et al.*, 2004]. [Torralba *et al.*, 2007, 2004] investigated how features could be shared across classes for multi-class detection. A hierarchy of parts was proposed by [Bouchard & Triggs, 2005] and [Epshtein & Ullman, 2005].

A.3.2 Bag-of-Words Models

Spatial information is deliberately thrown away in bag-of-words models. These take inspiration from the textual information retrieval community [Baeza-Yates & Ribeiro-Neto, 1999], and extend textual words to *visual* words. These were used by [Sivic & Zisserman, 2003] for efficiently searching long video sequences. Naïve Bayes and SVM classifiers were compared in [Csurka *et al.*, 2004]. Probabilistic latent semantic analysis (pLSA) [Hofmann, 2001] was used for visual categories in [Sivic *et al.*, 2005], and robust learning from Google image search results was achieved in [Fergus *et al.*, 2005]. A hierarchical Bayesian extension to pLSA, latent Dirichlet allocation [Blei *et al.*, 2003], was used for learning unsupervised image segmentations in [Russel *et al.*, 2006].

A.3.3 Modeling Context

The concept of a scene *gist* was used to model context for object detection in [Torralba *et al.*, 2003, 2005]. The framework of [Kumar & Hebert, 2005] modeled semantic context in a discriminative hierarchy. Hierarchical Bayesian models of scenes, objects, and parts were suggested by [Sudderth *et al.*, 2005].

A.4 Combined Recognition & Segmentation

Most recently, research has investigated the fundamental combination of object recognition with segmentation. The Implicit Shape Model [Leibe & Schiele, 2004; Leibe *et al.*, 2004; Leibe & Schiele, 2003] uses segmented image patches to detect and segment objects. OBJ CUT by [Kumar *et al.*, 2005] detected and segmented objects using a Markov random field combined with pictorial structures. Winn & Shotton addressed the detection and segmentation of par-

tially occluded objects in [Winn & Shotton, 2006]. This has been extended to recognizing three-dimensional objects in [Hoiem *et al.*, 2007].

APPENDIX B

BOOSTING ALGORITHMS

B.1 Introduction

For completeness, we include in this appendix a summary of the boosting algorithms employed for learning our recognition models. We first describe the binary Gentle AdaBoost algorithm [Friedman *et al.*, 2000], used in Chapters 2 and 4. We then detail the multi-class extension, Joint Boost, which was proposed in [Torralba *et al.*, 2004] and aims to share features between classes (e.g. blue is indicative of sky and water). Joint Boost was used for semantic segmentation in Chapter 3. Note that the abstract notation of this appendix differs very slightly from the concrete implementations used in the main text.

B.2 Gentle AdaBoost

The Gentle AdaBoost algorithm, from [Friedman *et al.*, 2000], is used to learn a classifier of the form

$$H(\mathbf{v}) = \sum_{m=1}^{M} h_m(\mathbf{v}) \,, \tag{B.1}$$

which takes a feature vector $\mathbf{v}$ and computes a strong classification value H as a sum of M *weak learners*. Each weak learner $h(\mathbf{v})$ contributes to the classification but individually need not be particularly discriminative. In the combination in (B.1) however, they form a powerful, discriminative classifier. The classification H can be reinterpreted as a posterior class probability using the logistic transformation:

$$P(c = +1|\mathbf{v}) = \frac{1}{1 + \exp(-H(\mathbf{v}))} \,, \tag{B.2}$$

with class $c \in \{+1, -1\}$.

Gentle AdaBoost is a particular algorithm for constructing the additive model of (B.1) in a greedy, iterative fashion. After $m-1$ rounds (iterations) of boosting, the next weak learner h_m is chosen from a pool $\mathcal{H}$ to minimize a weighted squared error functional on a set of training examples $i = 1, \ldots, N$:

$$h_m = \arg\min_{h \in \mathcal{H}} J_{\text{wse}}[h] \,, \tag{B.3}$$

$$J_{\text{wse}}[h] = \sum_{i=1}^{N} w_{i,m-1} \left(z_i - h(\mathbf{v}_i)\right)^2 \,, \tag{B.4}$$

where training example i consists of feature vector $\mathbf{v}_i$ and target label $z_i \in \{+1, -1\}$. The weights are

$$w_{i,m} = \exp(-z_i H_m(\mathbf{v}_i)) \,, \tag{B.5}$$

where $H_m = \sum_{m'=1}^{m} h_{m'}$ is the strong classifier up to round m. The weights represent the mis-classification of each training example after m rounds, so that the minimization in (B.3) gives more emphasis to poorly classified examples. The weights are all initialized as $w_{i,0} = 1$,[1] and can be efficiently updated after each iteration of boosting, maintaining (B.5) as an invariant, by

$$w_{i,m} = w_{i,m-1} \exp(-z_i h_m(\mathbf{v}_i)) \,. \tag{B.6}$$

B.2.1 Decision Stumps

The weak learners in pool $\mathcal{H}$ can in general take any form, but in this work we use the *decision stumps* of [Torralba *et al.*, 2004, 2007] which have the form

$$h(\mathbf{v}) = a[v_d > \theta] + b \,, \tag{B.7}$$

where v_d is the dth dimension of vector $\mathbf{v}$, and the binary indicator function $[\textit{condition}] = 1$ if *condition* is true, 0 otherwise. These decision stumps divide the feature space in two along an axis-aligned hyper-plane, and weight the two sides differently. The axis-alignment means that feature selection can be performed: by setting $M \ll D$, where D is the number of dimensions of $\mathbf{v}$, only a discriminative subset of feature dimensions (and thereby features) is

[1] The weights could in fact be initialized non-uniformly, and in this boosting variant need not sum to 1 since a constant scaling factor does not affect (B.3). Informal experiments suggested that initializing weights to normalize for imbalanced numbers of positive and negative training examples gave worse performance. This is likely due to the removal of the implicit and useful class priors that the classifier otherwise learns.

selected. At test time only those features that were selected need be evaluated. The ordering of the weak learners shows a general trend due to the weighting of the examples (B.5): early rounds select more general features, while later rounds concentrate on more specific troublesome examples. This ordering can be exploited in a cascade.

The decision stumps also have convenient analytical properties: for the minimization of (B.3), although a brute-force search is required for feature dimensions d and threshold θ (from a discrete set), given these values, a closed-form minimum for a and b exists:

$$b = \frac{\sum_i w_i z_i [v_{i,d} \leq \theta]}{\sum_i w_i [v_{i,d} \leq \theta]} , \tag{B.8}$$

$$a = \frac{\sum_i w_i z_i [v_{i,d} > \theta]}{\sum_i w_i [v_{i,d} > \theta]} - b , \tag{B.9}$$

where weights $w_i = w_{i,m-1}$ when minimizing (B.3) at round m.

B.3 Joint Boost

The Joint Boost algorithm is an extension of Gentle AdaBoost to multiple classes by Torralba *et al*. in [Torralba *et al.*, 2004], who use the novel insight that individual features can contribute towards the classification of several classes at once. This sharing of features across classes allows for classification with cost sub-linear in the number of classes, and leads to improved generalization; see [Torralba *et al.*, 2004]. Here, the classifier takes the form

$$H(c, \mathbf{v}) = \sum_{m=1}^{M} h_m(c, \mathbf{v}) , \tag{B.10}$$

which now additionally takes a class label $c \in \{1, \ldots, C\}$ as parameter (cf. (B.1)). The softmax or multi-class logistic transformation gives a class probability distribution:

$$P(c|\mathbf{v}) = \frac{1}{Z} \exp H(c, \mathbf{v}) \tag{B.11}$$

where $Z = \sum_c \exp H(c, \mathbf{v})$ normalizes the distribution.

The minimization to determine the optimal weak learner at each round becomes

$$h_m = \arg\min_{h \in \mathcal{H}} J_{\text{wse}}[h] , \tag{B.12}$$

$$J_{\text{wse}}[h] = \sum_{c=1}^{C} \sum_{i=1}^{N} w_{i,m-1}^{c} (z_i^c - h(c, \mathbf{v}_i))^2 , \tag{B.13}$$

where now there are C weights and targets for each example, and the superscript c denotes an index rather than an exponentiation. Target $z_i^c \in \{+1, -1\}$ is positive if example i is of class c, and negative otherwise.[2] The weights become

$$w_{i,m}^c = \exp(-z_i^c H_m(c, \mathbf{v}_i)) \ , \tag{B.14}$$

with the corresponding update equation

$$w_{i,m}^c = w_{i,m-1}^c \exp(-z_i^c h_m(c, \mathbf{v}_i)) \ . \tag{B.15}$$

B.3.1 Decision Stumps

The Joint Boost decision stumps take the form:

$$h(c, \mathbf{v}) = \begin{cases} a[v_d > \theta] + b & \text{if } c \in \mathcal{C}, \\ k^c & \text{otherwise.} \end{cases} \tag{B.16}$$

Here, $\mathcal{C} \subseteq \{1, \ldots, C\}$ is the set of classes between which the weak learner is shared. The constants k^c for $c \notin \mathcal{C}$ ensure that unequal numbers of training examples of each class do not adversely affect the learning procedure.

The parameters of each weak learner are therefore sharing set $\mathcal{C}$, the set of constants k^c, weights a and b, feature dimension d, and threshold θ. The set of all possible sharing sets is exponentially large, so we employ the quadratic-cost greedy approximation of [Torralba *et al.*, 2004]. This finds the optimal one-class sharing set, $\mathcal{C}_1 = \{c_1\}$, then, keeping the first class c_1 fixed, the optimal two-class sharing set, $\mathcal{C}_2 = \{c_1, c_2\}$, etc. Finally the best set $\mathcal{C}$ is chosen from $\{\mathcal{C}_1, \ldots, \mathcal{C}_C\}$. As with the binary Gentle AdaBoost algorithm, a brute-force search is then required to find d and θ, but given these, the remaining parameters come out

[2] In this work, we only allow one target to be positive for each example, except for the separable TextonBoost described in Section 3.4.2. Perhaps multiple targets could be also used to learn a hierarchical class structure, such as that cows and horses are both mammals.

analytically:

$$b = \frac{\sum_{c\in\mathcal{C}}\sum_i w_i^c z_i^c [v_{i,d} \leq \theta]}{\sum_{c\in\mathcal{C}}\sum_i w_i^c [v_{i,d} \leq \theta]} , \tag{B.17}$$

$$a = \frac{\sum_{c\in\mathcal{C}}\sum_i w_i^c z_i^c [v_{i,d} > \theta]}{\sum_{c\in\mathcal{C}}\sum_i w_i^c [v_{i,d} > \theta]} - b , \tag{B.18}$$

$$k^c = \frac{\sum_i w_i^c z_i^c}{\sum_i w_i^c} , \tag{B.19}$$

where weights $w_i^c = w_{i,m-1}^c$ when performing the minimization (B.12) at round m.

B.4 Optimizations

We briefly describe two optimizations of the above algorithms.

B.4.1 Search for θ

Brute-force search for θ from a discrete set Θ can be made efficient (giving identical results) by careful use of histograms of weighted feature responses: by treating Θ as an ordered set, histograms of feature values $v_{i,d}$ weighted appropriately by $w_i^c z_i^c$ and w_i^c, are built over bins corresponding to the thresholds in Θ; these histogram are accumulated to give the thresholded sums necessary for the direct calculation of a, b for all values of θ at once.

B.4.2 Randomization

A full search over all possible feature dimensions d at each round need not be performed. Instead, one can investigate only a small random fraction of feature dimensions. We investigate this approximation in Section 3.4.3.

Appendix C

Weizmann & Graz Datasets

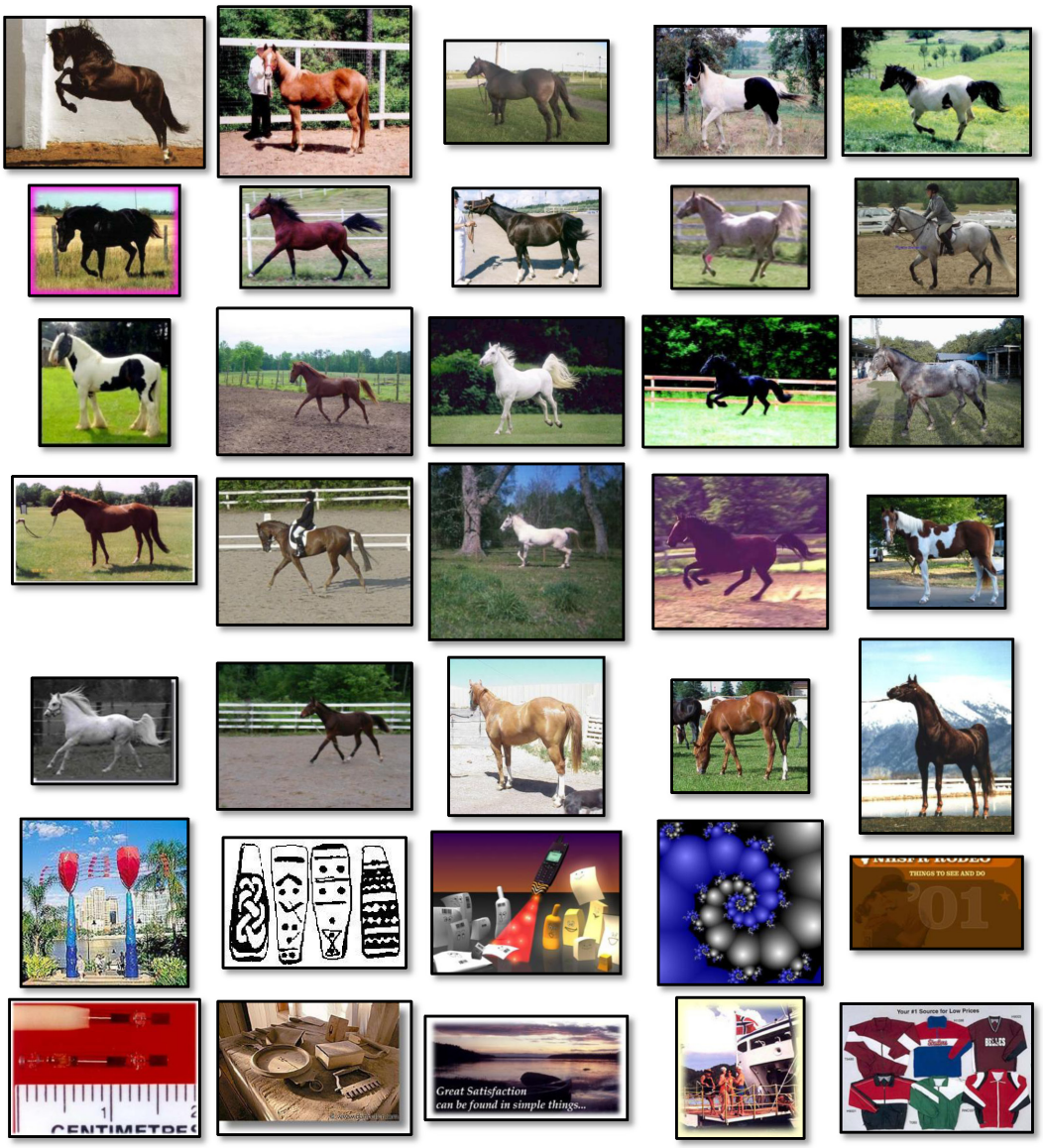

Figure C.1: **Example images from the multi-scale Weizmann horse dataset.** The dataset can be downloaded from [Shotton]. The bottom two rows contain example background images. This dataset was used in the evaluations of Chapters 2 and 4.

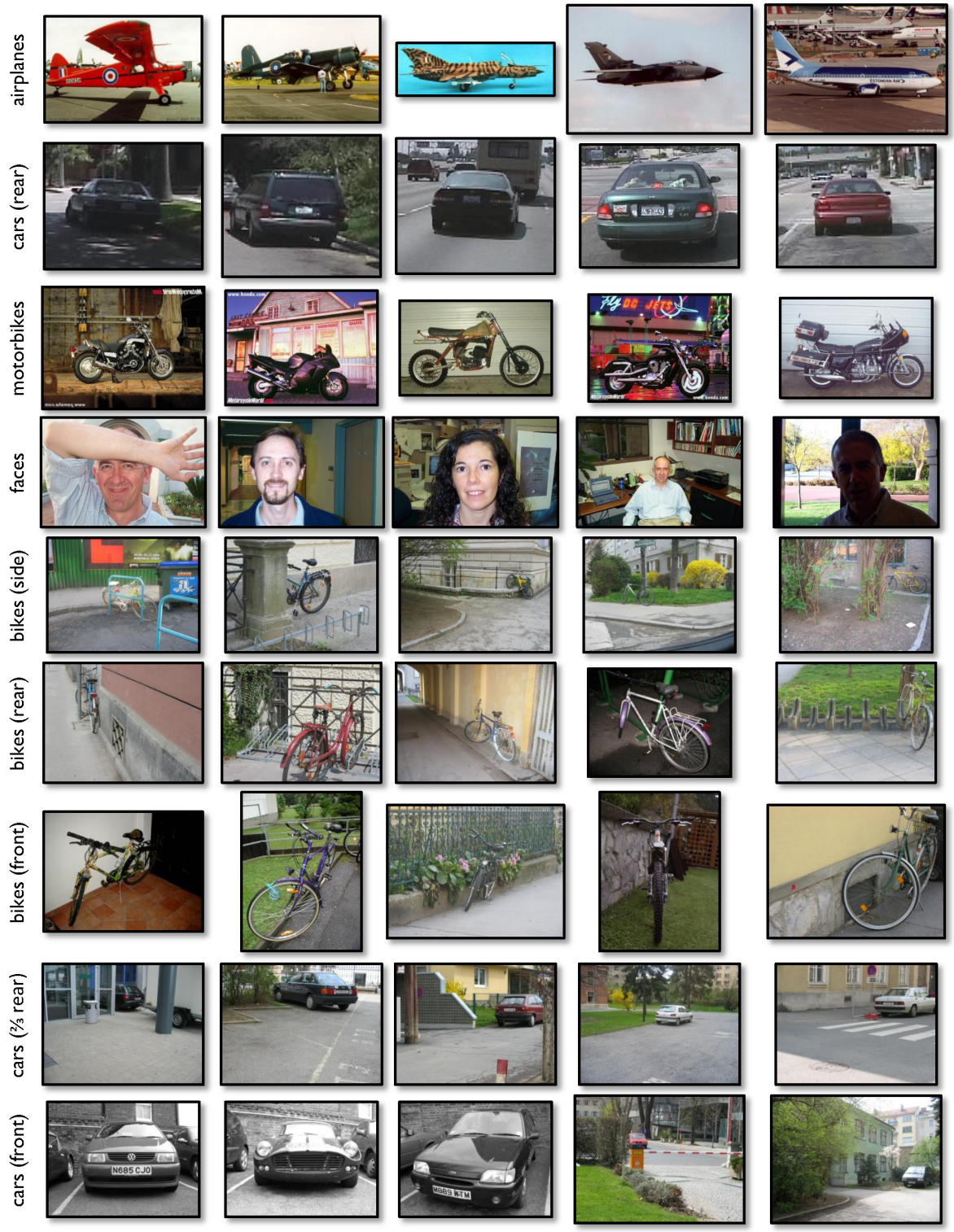

Figure C.2: **Example images from the Graz 17 class dataset from [Opelt *et al.*, 2006c].** This dataset was used in the evaluations of Chapters 2 and 4.

Figure C.3: **Example images from the Graz 17 class dataset from [Opelt *et al.*, 2006c].** This dataset was used in the evaluations of Chapters 2 and 4.

BIBLIOGRAPHY

AGARWAL, S., & ROTH, D. Learning a Sparse Representation for Object Detection. *Pages 113–130 of:* HEYDEN, A., SPARR, G., & JOHANSEN, P. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 2353. Springer. 2002.

ANDERSON, R, KINGSBURY, N, & FAUQUEUR, J. Rotation-invariant object recognition using edge-profile clusters. *In: Proc. European Conf. on Signal Processing (EUSIPCO)*. 2006 (Sept.).

BAEZA-YATES, R., & RIBEIRO-NETO, B. *Modern Information Retrieval*. ACM Press. 1999.

BALLARD, D.H. Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recognition*, **13**(2), 111–122. 1981.

BALUJA, S., & ROWLEY, H. A. Boosting Sex Identification Performance. *Pages 1508–1513 of: AAAI*. 2005.

BARROW, H.G., TENENBAUM, J.M., BOLLES, R.C., & WOLF, H.C. Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching. *Pages 659–663 of: Proc. 5th Int. Joint Conf. Artificial Intelligence*. 1977.

BAUMBERG, A. Reliable Feature Matching across Widely Separated Views. *Pages 774–781 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2000 (June).

BEAL, M.J. 2003. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, Gatsby Computational Neuroscience Unit, University College London.

BEIS, J.S., & LOWE, D.G. Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. *Pages 1000–1006 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*. 1997 (June).

BELONGIE, S., MALIK, J., & PUZICHA, J. Shape Matching and Object Recognition Using Shape Contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **24**(24), 509–522. 2002.

BERG, A.C., & MALIK, J. Geometric Blur for Template Matching. *Pages 607–614 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2001 (Dec.).

BERG, A.C., BERG, T.L., & MALIK, J. Shape Matching and Object Recognition using Low Distortion Correspondences. *Pages 26–33 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2005 (June).

BIEDERMAN, I., & JU, G. Surface vs. Edge-Based Determinants of Visual Recognition. *Cognitive Psychology*, **20**(1), 38–64. 1988.

BISHOP, C.M. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc. 2006.

BLAKE, A., ROTHER, C., BROWN, M., PEREZ, P., & TORR, P.H.S. Interactive Image Segmentation Using an Adaptive GMMRF Model. *Pages 428–441 of:* PAJDLA, T., & MATAS, J. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3021. Prague, Czech Republic: Springer. 2004.

BLEI, D.M., NG, A.Y., & JORDAN, M. I. Latent Dirichlet Allocation. *J. Machine Learning Research*, **3**(Jan.), 993–1022. 2003.

BORENSTEIN, E., & MALIK, J. Shape Guided Object Segmentation. *Pages 969–976 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2006 (June).

BORENSTEIN, E., & ULLMAN, S. Class-Specific, Top-Down Segmentation. *Pages 109–124 of:* HEYDEN, A., SPARR, G., & JOHANSEN, P. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 2351. Springer. 2002.

BORENSTEIN, E., & ULLMAN, S. Learning to Segment. *Pages 315–328 of:* PAJDLA, T., & MATAS, J. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3023. Springer. 2004.

BORENSTEIN, E., SHARON, E., & ULLMAN, S. Combining Top-down and Bottom-up Segmentations. *Page 46 of: IEEE Workshop on Perceptual Organization in Computer Vision*, vol. 4. 2004.

BORGEFORS, G. Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **10**(6), 849–865. 1988.

BOUCHARD, G.G., & TRIGGS, B. Hierarchical part-based visual object categorization. *Pages 710–715 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2005 (June).

BOYKOV, Y., & JOLLY, M.-P. Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images. *Pages 105–112 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 2001 (July).

BOYKOV, Y., VEKSLER, O., & JOLLY, M.-P. Fast Approximate Energy Minimization via Graph Cuts. *Pages 377–384 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 1999 (Sept.).

BOYKOV, Y., VEKSLER, O., & ZABIH, R. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **23**(11), 1222–1239. 2001.

BROWN, M., SZELISKI, R., & WINDER, S. Multi-Image Matching using Multi-Scale Oriented Patches. *Pages 510–517 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2005 (June).

BURL, M.C., WEBER, M., & PERONA, P. A Probabilistic Approach to Object Recognition Using Local Photometry and Global Geometry. *Pages 628–641 of:* BURKHARDT, H., & NEUMANN, B. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 1407. Springer. 1998.

CALTECH. The Caltech Database.
`http://www.robots.ox.ac.uk/~vgg/data.html`.

CALTECH 101. The Caltech 101 Database.
`http://www.vision.caltech.edu/Image_Datasets/Caltech101/Caltech101.html`.

CANNY, J. A Computational Approach to Edge Detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **8**(6), 679–698. 1986.

CARMICHAEL, O., & HEBERT, M. Shape-Based Recognition of Wiry Objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **26**(12), 1537–1552. 2004.

CHEUNG, V., FREY, B.J., & JOJIC, N. Video epitomes. *Pages 42–49 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2005 (June).

COMANICIU, D., & MEER, P. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **24**(5), 603–619. 2002.

COOTES, T.F., & TAYLOR, C.J. Statistical models of appearance for medical image analysis and computer vision. *Pages 236–248 of:* SONKA, M., & HANSON, K.M. (eds), *Proc. SPIE Medical Imaging*, vol. 4322. 2001 (July).

CRIMINISI, A., PEREZ, P., & TOYAMA, K. Region Filling and Object Removal by Exemplar-Based Inpainting. *IEEE Trans. on Image Processing*, **13**(9), 1200–1212. 2004.

CRIMINISI, A., SHOTTON, J., BLAKE, A., ROTHER, C., & TORR, P.H.S. Efficient Dense Stereo with Occlusions for New View-Synthesis by Four-State Dynamic Programming. *Int. J. Computer Vision*, **71**(1), 89–110. 2007.

CSURKA, G., DANCE, C.R., FAN, L., WILLAMOWSKI, J., & BRAY, C. Visual Categorization with Bags of Keypoints. *In: ECCV International Workshop on Statistical Learning in Computer Vision*. 2004.

DAUGMAN, J.G. Two-dimensional spectral analysis of cortical receptive field profiles. *Vision Res.*, **20**(10), 847–856. 1980.

DAUGMAN, J.G. Demodulation by complex-valued wavelets for stochastic pattern recognition. *Int. J. Wavelets, Multi-resolution and Image Processing*, **1**(1), 1–17. 2003.

DAVISON, A.J., REID, I.D., & MOLTON, N.D. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **29**(6), 1–16. 2007.

DOLLÁR, P., TU, Z., & BELONGIE, S. Supervised Learning of Edges and Object Boundaries. *Pages 1964–1971 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. 2006.

DORKÓ, G., & SCHMID, C. Selection of Scale-Invariant Parts for Object Class Recognition. *Pages 634–639 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 2003 (Oct.).

DORKÓ, G., & SCHMID, C. *Object Class Recognition Using Discriminative Local Features*. Tech. rept. RR-5497. INRIA - Rhone-Alpes. 2005 (February).

DUYGULU, P., BARNARD, K., DE FREITAS, N., & FORSYTH, D. Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. *Pages 97–112 of:* HEYDEN, A., SPARR, G., & JOHANSEN, P. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 2353. Springer. 2002.

ELKAN, C. Using the Triangle Inequality to Accelerate k-Means. *Pages 147–153 of: Proc. Int. Conf. on Machine Learning*. 2003.

EPSHTEIN, B., & ULLMAN, S. Feature Hierarchies for Object Classification. *Pages 220–227 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 2005 (Oct.).

FEI-FEI, L., FERGUS, R., & PERONA, P. One-Shot Learning of Object Categories. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **28**(4), 594–611. 2006.

FELZENSZWALB, P.F. Learning Models for Object Recognition. *Pages 1056–1062 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2001 (Dec.).

FELZENSZWALB, P.F., & HUTTENLOCHER, D.P. Efficient Matching of Pictorial Structures. *Pages 66–73 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. 2000.

FELZENSZWALB, P.F., & HUTTENLOCHER, D.P. *Distance Transforms of Sampled Functions*. Tech. rept. Cornell. 2004.

FELZENSZWALB, P.F., & HUTTENLOCHER, D.P. Pictorial Structures for Object Recognition. *Int. J. Computer Vision*, **61**(1), 55–79. 2005.

FERGUS, R., PERONA, P., & ZISSERMAN, A. Object Class Recognition by Unsupervised Scale-Invariant Learning. *Pages 264–271 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. 2003 (June).

FERGUS, R., PERONA, P., & ZISSERMAN, A. A Visual Category Filter for Google Images. *Pages 242–256 of:* PAJDLA, T., & MATAS, J. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3021. Springer. 2004.

FERGUS, R., FEI-FEI, L., PERONA, P., & ZISSERMAN, A. Learning Object Categories from Google's Image Search. *Pages 1816–1823 of: Proc. Int. Conf. on Computer Vision*, vol. 2. 2005 (Oct.).

FERRARI, V., TUYTELAARS, T., & VAN GOOL, L.J. Simultaneous Object Recognition and Segmentation by Image Exploration. *Pages 40–54 of:* PAJDLA, T., & MATAS, J. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3021. Prague, Czech Republic: Springer. 2004.

FERRARI, V., FEVRIER, L., JURIE, F., & SCHMID, C. *Groups of Adjacent Contour Segments for Object Detection*. Tech. rept. 5980. INRIA. 2006a (September).

FERRARI, V., TUYTELAARS, T., & VAN GOOL, L. Object Detection by Contour Segment Networks. *Pages 14–28 of:* LEONARDIS, A., BISCHOF, H., & PINZ, A. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3953. Springer. 2006b.

FISCHLER, M.A., & ELSCHLAGER, R.A. The Representation and Matching of Pictorial Structures. *IEEE Trans. on Computers*, **22**(1), 67–92. 1973.

FÖRSTNER, W., & GÜLCH, E. A fast operator for detection and precise location of distinct points, corners and centres of circular features. *Pages 149–155 of: ISPRS Intercommission Workshop*. 1987 (June).

FREEMAN, W.T. 1992. *Steerable Filters and Analysis of Image Structure*. Ph.D. thesis, MIT.

FRIEDMAN, J., HASTIE, T., & TIBSHIRANI, R. Additive Logistic Regression: a Statistical View of Boosting. *Annals of Statistics*, **28**(2), 337–407. 2000.

GAVRILA, D.M. Multi-Feature Hierarchical Template Matching Using Distance Transforms. *Pages 439–444 of: Proc. Int. Conf. on Pattern Recognition*, vol. 1. 1998 (Aug.).

GAVRILA, D.M. Pedestrian Detection from a Moving Vehicle. *Pages 37–49 of:* VERNON, D. (ed), *Proc. European Conf. on Computer Vision*, vol. LNCS 1843. Springer. 2000.

GEMAN, S., & GEMAN, D. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **6**(6), 721–741. 1984.

GOOGLE. Google.
`http://www.google.com/`.

HARRIS, C., & STEPHENS, M. A combined corner and edge detector. *Pages 147–151 of: Proceedings of the 4th ALVEY vision conference*. 1988.

HE, X., ZEMEL, R.S., & CARREIRA-PERPIÑÁN, M.Á. Multiscale Conditional Random Fields for Image Labeling. *Pages 695–702 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. 2004 (June).

HE, X., ZEMEL, R.S., & RAY, D. Learning and Incorporating Top-Down Cues in Image Segmentation. *Pages 338–351 of:* LEONARDIS, A., BISCHOF, H., & PINZ, A. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3951. Springer. 2006.

HOFMANN, T. Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Machine Learning*, **42**(1-2), 177–196. 2001.

HOIEM, D., EFROS, A.A., & HEBERT, M. Geometric Context from a Single Image. *Pages 654–661 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 2005 (Oct.).

HOIEM, D., ROTHER, C., & WINN, J. 3D LayoutCRF for Multi-View Object Class Recognition and Segmentation. *In: Proc. IEEE Conf. Computer Vision and Pattern Recognition*. 2007.

HUTTENLOCHER, D.P., & RUCKLIDGE, W.J. *A Multi-Resolution Technique for Comparing Images Using the Hausdorff Distance*. Tech. rept. TR 92-1321. Department of Computer Science, Cornell University. 1992 (December).

IOFFE, S., & FORSYTH, D. Mixtures of Trees for Object Recognition. *Pages 180–185 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. 2001 (Dec.).

JOHNSON, M., BROSTOW, G., SHOTTON, J., ARANDJELOVIC, O., KWATRA, V., & CIPOLLA, R. Semantic Photo Synthesis. *Computer Graphics Forum*, **25**(3), 407–413. 2006.

JOJIC, N., FREY, B.J., & KANNAN, A. Epitomic analysis of appearance and shape. *Pages 34–41 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 2003 (Oct.).

JULESZ, B. Textons, the elements of texture perception, and their interactions. *Nature*, **290**(5802), 91–97. 1981.

JURIE, F., & SCHMID, C. Scale-invariant shape features for recognition of object categories. *Pages 90–96 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. 2004 (June).

JURIE, F., & TRIGGS, B. Creating Efficient Codebooks for Visual Recognition. *Pages 604–610 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 2005 (Oct.).

KADIR, T., & BRADY, M. Saliency, Scale and Image Description. *Int. J. Computer Vision*, **45**(2), 83–105. 2001.

KADIR, T., BOUKERROUI, D., & BRADY, M. *An analysis of the Scale Saliency algorithm*. Tech. rept. University of Oxford. 2003.

KADIR, T., ZISSERMAN, A., & BRADY, M. An Affine Invariant Salient Region Detector. *Pages 228–241 of:* PAJDLA, T., & MATAS, J. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3021. Springer. 2004.

KANNAN, A., WINN, J., & ROTHER, C. Clustering appearance and shape by learning jigsaws. *In: Advances in Neural Information Processing Systems*. 2006.

KAPOOR, A., & WINN, J. Located Hidden Random Fields: Learning Discriminative Parts for Object Detection. *Pages 302–315 of:* LEONARDIS, A., BISCHOF, H., & PINZ, A. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3953. Springer. 2006.

KIENZLE, W., WICHMANN, F.A., SCHÖLKOPF, B., & FRANZ, M.O. A Nonparametric Approach to Bottom-Up Visual Saliency. *In: Advances in Neural Information Processing Systems*. 2006.

KINGSBURY, N.G. Complex wavelets for shift invariant analysis and filtering of signals. *J. Applied and Computational Harmonic Analysis*, **10**(3), 234–253. 2001.

KOHLI, P., & TORR, P.H.S. Efficiently Solving Dynamic Markov Random Fields Using Graph Cuts. *Pages 922–929 of: Proc. Int. Conf. on Computer Vision*, vol. 2. 2005 (Oct.).

KOLMOGOROV, V., & ZABIH, R. What Energy Functions Can be Minimized Via Graph Cuts? *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **26**(2), 147–159. 2004.

KONISHI, S., & YUILLE, A. L. Statistical Cues for Domain Specific Image Segmentation with Performance Analysis. *Pages 125–132 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2000 (June).

KUMAR, M.P., TORR, P.H.S., & ZISSERMAN, A. Extending Pictorial Structures for Object Recognition. *In: Proc. British Machine Vision Conference*. 2004.

KUMAR, M.P., TORR, P.H.S., & ZISSERMAN, A. OBJ CUT. *Pages 18–25 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2005 (June).

KUMAR, S., & HEBERT, M. Discriminative Random Fields: A Discriminative Framework for Contextual Interaction in Classification. *Pages 1150–1157 of: Proc. Int. Conf. on Computer Vision*, vol. 2. 2003 (Oct.).

KUMAR, S., & HEBERT, M. A Hierarchical Field Framework for Unified Context-Based Classification. *Pages 1284–1291 of: Proc. Int. Conf. on Computer Vision*, vol. 2. 2005 (Oct.).

LAFFERTY, J., MCCALLUM, A., & PEREIRA, F. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Pages 282–289 of: Proc. Int. Conf. on Machine Learning*. 2001.

LAPTEV, I., & LINDEBERG, T. Space-time interest points. *Pages 432–439 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 2003 (Oct.).

LEIBE, B., & SCHIELE, B. Interleaved Object Categorization and Segmentation. *Pages 264–271 of: Proc. British Machine Vision Conference*, vol. II. 2003.

LEIBE, B., & SCHIELE, B. Scale Invariant Object Categorization Using a Scale-Adaptive Mean-Shift Search. *Pages 145–153 of: DAGM'04: 26th Pattern Recognition Symposium*. 2004 (June).

LEIBE, B., LEONARDIS, A., & SCHIELE, B. Combined Object Categorization and Segmentation with an Implicit Shape Model. *In: ECCV'04 Workshop on Statistical Learning in Computer Vision*. 2004 (May).

LEIBE, B., SEEMANN, E., & SCHIELE, B. Pedestrian Detection in Crowded Scenes. *Pages 878–885 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2005 (June).

LEPETIT, V., LAGGER, P., & FUA, P. Randomized Trees for Real-Time Keypoint Recognition. *Pages 775–781 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. 2005 (June).

LEUNG, T. Texton Correlation for Recognition. *Pages 203–214 of:* PAJDLA, T., & MATAS, J. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3021. Springer. 2004.

LEUNG, T., & MALIK, J. Representing and Recognizing the Visual Appearance of Materials using Three-Dimensional Textons. *Int. J. Computer Vision*, **43**(1), 29–44. 2001.

LEVIN, A., & WEISS, Y. Learning to Combine Bottom-Up and Top-Down Segmentation. *Pages 581–594 of:* LEONARDIS, A., BISCHOF, H., & PINZ, A. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3954. Springer. 2006.

LINDEBERG, T., & GÅRDING, J. Shape-Adapted Smoothing in Estimation of 3D Depth Cues from Affine Distortions of Local 2D Brightness Structure. *Image and Vision Computing*, **15**(6), 415–434. 1997.

LIVE. Live Search.
`http://www.live.com/`.

LOWE, D.G. Object Recognition from Local Scale-Invariant Features. *Pages 1150–1157 of: Proc. Int. Conf. on Computer Vision*, vol. 2. 1999 (Sept.).

LOWE, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Computer Vision*, **60**(2), 91–110. 2004.

MALIK, J., & PERONA, P. Preattentive texture discrimination with early vision mechanisms. *J. Optical Society of America A*, **12**(7), 629–639. 1990.

MALIK, J., BELONGIE, S., LEUNG, T., & SHI, J. Contour and Texture Analysis for Image Segmentation. *Int. J. Computer Vision*, **43**(1), 7–27. 2001.

MARR, D. *Vision: A computational investigation into the human representation and processing of visual information*. W. H. Freeman. 1982.

MARTIN, D.R., FOWLKES, C.C., & MALIK, J. Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **26**(5), 530–549. 2004.

MATAS, J., CHUM, O., URBAN, M., & PAJDLA, T. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. *Pages 384–393 of: Proc. British Machine Vision Conference*, vol. 1. 2002.

MIKOLAJCZYK, K., & SCHMID, C. Indexing Based on Scale Invariant Interest Points. *Pages 525–531 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 2001 (July).

MIKOLAJCZYK, K., & SCHMID, C. An Affine Invariant Interest Point Detector. *Pages 128–142 of:* HEYDEN, A., SPARR, G., & JOHANSEN, P. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 2350. Springer. 2002.

MIKOLAJCZYK, K., & SCHMID, C. A performance evaluation of local descriptors. *Pages 257–263 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. 2003 (June).

MIKOLAJCZYK, K., & SCHMID, C. Scale and affine invariant interest point detectors. *Int. J. Computer Vision*, **60**(1), 63–86. 2004.

MIKOLAJCZYK, K., ZISSERMAN, A., & SCHMID, C. Shape recognition with edge-based features. *Pages 779–788 of: Proc. British Machine Vision Conference*, vol. 2. 2003 (Sept.).

MIKOLAJCZYK, K., SCHMID, C., & ZISSERMAN, A. Human detection based on a probabilistic assembly of robust part detectors. *Pages 69–82 of:* PAJDLA, T., & MATAS, J. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3021. Springer. 2004.

MIKOLAJCZYK, K., LEIBE, B., & SCHIELE, B. Local Features for Object Class Recognition. *Pages 1792–1799 of: Proc. Int. Conf. on Computer Vision*, vol. 2. 2005 (Oct.).

MSRC 21. The Microsoft Research Cambridge 21 Class Database.
`http://research.microsoft.com/vision/cambridge/recognition/`.

NAVARATNAM, R., THAYANANTHAN, A., TORR, P.H.S., & CIPOLLA, R. Hierarchical Part-Based Human Body Pose Estimation. *In: Proc. British Machine Vision Conference*. 2005.

NELSON, R.C., & SELINGER, A. A Cubist Approach to Object Recognition. *Pages 614–621 of: Proc. Int. Conf. on Computer Vision*. 1998 (Jan.).

OLSON, C.F., & HUTTENLOCHER, D.P. Automatic Target Recognition by Matching Oriented Edge Pixels. *IEEE Trans. on Image Processing*, **6**(1), 103–113. 1997.

OPELT, A., PINZ, A., & ZISSERMAN, A. A Boundary-Fragment-Model for Object Detection. *Pages 575–588 of:* LEONARDIS, A., BISCHOF, H., & PINZ, A. (eds), *Proc. European Conf. on Computer Vision*, vol. 2. Graz, Austria: Springer. 2006a.

OPELT, A., PINZ, A., & ZISSERMAN, A. Fusing shape and appearance information for object category detection. *In: Proc. British Machine Vision Conference*. 2006b.

OPELT, A., PINZ, A., & ZISSERMAN, A. Incremental learning of object detectors using a visual shape alphabet. *Pages 3–10 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2006c (June).

PEARL, J. *Probabilistic Reasoning in Intelligent Systems*. San Mateo: Morgan Kaufmann. 1988.

PEEKABOOM. Peekaboom.
`http://www.peekaboom.org/`.

PORIKLI, F. M. Integral Histogram: A Fast Way To Extract Histograms in Cartesian Spaces. *Pages 829–836 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2005 (June).

ROTHER, C., KOLMOGOROV, V., & BLAKE, A. GrabCut - Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Transactions on Graphics*, **23**(3), 309–314. 2004.

ROTHER, C., BORDEAUX, L., HAMADI, Y., & BLAKE, A. AutoCollage. *ACM Transactions on Graphics*, **25**(3), 847–852. 2006.

RUSSEL, B.C., TORRALBA, A., MURPHY, K.P., & FREEMAN, W.T. *LabelMe: a database and web-based tool for image annotation*. Tech. rept. 25. MIT AI Lab. 2005 (September).

RUSSEL, B.C., EFROS, A.A., SIVIC, J., FREEMAN, W.T., & ZISSERMAN, A. Using Multiple Segmentations to Discover Objects and their Extent in Image Collections. *Pages 1605–1614 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. 2006 (June).

SAVARESE, S., WINN, J., & CRIMINISI, A. Discriminative Object Class Models of Appearance and Shape by Correlations. *Pages 2033–2040 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. 2006 (June).

SCHAFFALITZKY, F., & ZISSERMAN, A. Automated Scene Matching in Movies. *Pages 186–197 of: Proc. Int. Conf. on Image and Video Retrieval*, vol. LNCS 2383. 2002a.

SCHAFFALITZKY, F., & ZISSERMAN, A. Multi-view Matching for Unordered Image Sets, or 'How Do I Organize My Holiday Snaps?'. *Pages 414–431 of:* HEYDEN, A., SPARR, G., & JOHANSEN, P. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 2350. Springer. 2002b.

SCHMID, C., & MOHR, R. Local Grayvalue Invariants for Image Retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **19**(5), 530–535. 1997.

SCHNEIDERMAN, H., & KANADE, T. A Statistical Method for 3D Object Detection Applied to Faces and Cars. *Pages 746–751 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2000 (June).

SCHNEIDERMAN, H., & KANADE, T. Object Detection Using the Statistics of Parts. *Int. J. Computer Vision*, **56**(3), 151–177. 2004.

SE, S., LOWE, D.G., & LITTLE, J. Vision-based Global Localization and Mapping for Mobile Robots. *IEEE Trans. on Robotics*, **21**(3), 364–375. 2005.

SHI, J., & MALIK, J. Normalized Cuts and Image Segmentation. *Pages 731–737 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*. 1997 (June).

SHOTTON. Jamie Shotton's Homepage.
`http://mi.eng.cam.ac.uk/~jdjs2/`.

SHOTTON, J., BLAKE, A., & CIPOLLA, R. Contour-Based Learning for Object Detection. *Pages 503–510 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 2005 (Oct.).

SHOTTON, J., WINN, J., ROTHER, C., & CRIMINISI, A. *TextonBoost*: Joint Appearance, Shape and Context Modeling for Multi-Class Object Recognition and Segmentation. *Pages 1–15 of:* LEONARDIS, A., BISCHOF, H., & PINZ, A. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3951. Springer. 2006.

SIVIC, J., & ZISSERMAN, A. Video Google: A text retrieval approach to object matching in videos. *Pages 1470–1477 of: Proc. Int. Conf. on Computer Vision*, vol. 2. 2003 (Oct.).

SIVIC, J., RUSSEL, B.C., EFROS, A.A., ZISSERMAN, A., & FREEMAN, W.T. Discovering objects and their localization in images. *Pages 370–377 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 2005 (Oct.).

STENGER, B., THAYANANTHAN, A., TORR, P.H.S., & CIPOLLA, R. Filtering using a tree-based estimator. *Pages 1063–1070 of: Proc. Int. Conf. on Computer Vision*, vol. 2. 2003 (Oct.).

SUDDERTH, E.B., TORRALBA, A., FREEMAN, W.T., & WILLSKY, A.S. Learning Hierarchical Models of Scenes, Objects, and Parts. *Pages 1331–1338 of: Proc. Int. Conf. on Computer Vision*, vol. 2. 2005 (Oct.).

SUTTON, C., & MCCALLUM, A. Piecewise Training of Undirected Models. *In: Proc. Conf. on Uncertainty in Artificial Intelligence*. 2005.

THAYANANTHAN, A., NAVARATNAM, R., TORR, P.H.S., & CIPOLLA, R. Likelihood Models for Template Matching Using the PDF Projection Theorem. *In: Proc. British Machine Vision Conference*. 2004 (September).

TORRALBA, A., MURPHY, K.P., FREEMAN, W.T., & RUBIN, M.A. Context-Based Vision System for Place and Object Recognition. *Pages 273–280 of: Proc. Int. Conf. on Computer Vision*, vol. 2. 2003 (Oct.).

TORRALBA, A., MURPHY, K.P., & FREEMAN, W.T. Sharing Features: Efficient Boosting Procedures for Multiclass Object Detection. *Pages 762–769 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. 2004 (June).

TORRALBA, A., MURPHY, K.P., & FREEMAN, W.T. Contextual Models for Object Detection using Boosted Random Fields. *Pages 1401–1408 of: Advances in Neural Information Processing Systems*. 2005.

TORRALBA, A., MURPHY, K.P., & FREEMAN, W.T. Sharing Features for Multiclass and Multiview Object Detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **29**(5). 2007.

TOYAMA, K., & BLAKE, A. Probabilistic Tracking with Exemplars in a Metric Space. *Int. J. Computer Vision*, **48**(1), 9–19. 2002.

TU, Z., & YUILLE, A.L. Shape Matching and Recognition – Using Generative Models and Informative Features. *Pages 195–209 of:* PAJDLA, T., & MATAS, J. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 3023. Springer. 2004.

TU, Z., CHEN, X., YUILLE, A.L., & ZHU, S.C. Image parsing: unifying segmentation, detection, and recognition. *Pages 18–25 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 2003 (Oct.).

TUYTELAARS, T., & VAN GOOL, L.J. Wide Baseline Stereo Matching based on Local, Affinely Invariant Regions. *In:* MIRMEHDI, M., & THOMAS, B.T. (eds), *Proc. British Machine Vision Conference*. 2000 (Sept.).

ULLMAN, S., SALI, E., & VIDAL-NAQUET, M. A Fragment-Based Approach to Object Representation and Classification. *Pages 85–102 of: Proc. 4th Int. Workshop on Visual Form*, vol. LNCS 2059. Capri, Italy: Springer-Verlag. 2001.

VAN GOOL, L.J., MOONS, T., & UNGUREANU, D. Affine/Photometric Invariants for Planar Intensity Patterns. *Pages 642–651 of:* BUXTON, B.F., & CIPOLLA, R. (eds), *Proc. European Conf. on Computer Vision*, vol. LNCS 1064. Springer. 1996.

VARMA, M., & ZISSERMAN, A. A Statistical Approach to Texture Classification from Single Images. *Int. J. Computer Vision*, **62**(1-2), 61–81. 2005.

VASCONCELOS, M., CARNEIRO, G., & VASCONCELOS, N. Weakly Supervised Top-down Image Segmentation. *Pages 1001–1006 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2006 (June).

VIOLA, P., & JONES, M.J. Rapid Object Detection using a Boosted Cascade of Simple Features. *Pages 511–518 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2001 (Dec.).

VOC. The Visual Object Classes Challenge.
`http://www.pascal-network.org/challenges/VOC/`.

WEBER, M., WELLING, M., & PERONA, P. Unsupervised Learning of Models for Recognition. *Pages 18–32 of:* VERNON, D. (ed), *Proc. European Conf. on Computer Vision*, vol. LNCS 1842. Springer. 2000.

WEIZMANN. The Weizmann Horse Database.
`http://www.msri.org/people/members/eranb/`.

WINN, J., & JOJIC, N. LOCUS: Learning Object Classes with Unsupervised Segmentation. *Pages 756–763 of: Proc. Int. Conf. on Computer Vision*, vol. 1. 2005 (Oct.).

WINN, J., & SHOTTON, J. The Layout Consistent Random Field for Recognizing and Segmenting Partially Occluded Objects. *Pages 37–44 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1. 2006 (June).

WINN, J., CRIMINISI, A., & MINKA, T. Categorization by Learned Universal Visual Dictionary. *Pages 1800–1807 of: Proc. Int. Conf. on Computer Vision*, vol. 2. 2005 (Oct.).

YAHOO. Yahoo!
`http://www.yahoo.com/`.

YEDIDIA, J.S., FREEMAN, W.T., & WEISS, Y. *Understanding belief propagation and its generalizations*. Morgan Kaufmann Publishers Inc. 2003.

YIN, P., CRIMINISI, A., WINN, J., & ESSA, I. Tree Based Classifiers for Bilayer Video Segmentation. *In: Proc. IEEE Conf. Computer Vision and Pattern Recognition*. 2007.

ZHANG, J., MARSZAŁEK, M., LAZEBNIK, S., & SCHMID, C. *Local Features and Kernels for Classificaiton of Texture and Object Categories: A Comprehensive Study*. Tech. rept. 5737. INRIA. 2005a.

ZHANG, W., YU, B., ZELINSKY, G.J., & SAMARAS, D. Object Class Recognition Using Multiple Layer Boosting with Heterogeneous Features. *Pages 323–330 of: Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2. 2005b (June).

ZHENG, S., TU, Z., & YUILLE, A.L. Detecting Object Boundaries Using Low-, Mid- and High-level Information. *In: Proc. IEEE Conf. Computer Vision and Pattern Recognition*. 2007.

ZHU, X., & GHAHRAMANI, Z. *Learning from labeled and unlabeled data with label propagation*. Tech. rept. CMU-CALD-02-107. Carnegie Mellon University. 2002.