Semantic Segmentation and Image Search

MATTHEW ALASTAIR JOHNSON

Hughes Hall

UNIVERSITY OF CAMBRIDGE

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

April 2008

ABSTRACT

Understanding the meaning behind visual data is increasingly important as the quantity of digital images in circulation explodes, and as computing in general and the Internet in specific shifts quickly towards an increasingly visual presentation of data. However, the remarkable amount of variance inside categories (*e.g.* different kinds of chairs) combined with the occurrence of similarity between categories (*e.g.* similar breeds of cats and dogs) makes this problem incredibly difficult to solve. In particular, the *semantic segmentation* of images into contiguous regions of similar interpretation combines the difficulties of object recognition and image segmentation to result in a problem of great complexity, yet great reward. This thesis proposes a novel solution to the problem of semantic segmentation, and explores its application to image search and retrieval.

Our primary contribution is a new image information processing tool: the *semantic texton forest*. We use semantic texton forests to perform (i) semantic segmentation of images and (ii) image categorization, achieving state-of-the-art results for both on two challenging datasets. We then apply this to the problem of image search and retrieval, resulting in the Palette Search system. With Palette Search, the user is able to search for the first time using *Query by Semantic Composition*, in which he communicates both what he wants in the result image and where he wants it.

Keywords: Computer, Vision, Object, Recognition, Image, Segmentation, Semantic, Web, Search, Retrieval, Composition, QBSC

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

None of this would have been possible without the guidance and aid of my inestimable supervisor Professor Roberto Cipolla. I would never have entered into computer vision research without the encouragement of Dr. Kobus Barnard, to whom goes my lasting thanks. The ideas contained within are the direct result of countless conversations and fruitful collaborations with Jamie Shotton, Gabriel Brostow, and John Winn, to whose Musketeers I have attempted to play D'Artagnan, and who have been true friends and valued colleagues of the highest order.

The research in this thesis was funded in part by the Toshiba Corporation, but mostly by the Louise Blouin Foundation, whose founder Louise Thérèse Blouin MacBain possesses my profound gratitude.

This thesis is dedicated to my dearest Libbi Ilyse, for all time first and foremost in my heart, my sustainer and my joy. Equally, to my loving father Graham, whose insistent encouragement sent me on my way, and my remarkable mother Eleanor, whose constant faith enabled me to finish.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

The inventor of the World Wide Web, Sir Timothy John Berners-Lee, spoke of a *Semantic Web* at the end of the last millennium as the next stage of its evolution:

> "I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The intelligent agents people have touted for ages will finally materialize."[7]

However, in a recent article written with colleagues he notes that 'This simple idea, however, remains largely unrealized' [88]. A fundamental part of this vision is the ability to understand the meaning behind the content of the web. The manner in which one performs a text query in a search engine is a good indicator of the current philosophy of the web. For example, if you want to find out about great horror movies of the 1950s, you type "great horror movies of the 1950s" into the query field of your favorite search website, and receive a list of pages which have all or some of those exact words in them. Due to some clever inference on the part of the information retrieval engine which underlies the search website which you are using, the pages at the top of the list are those pages which people have found

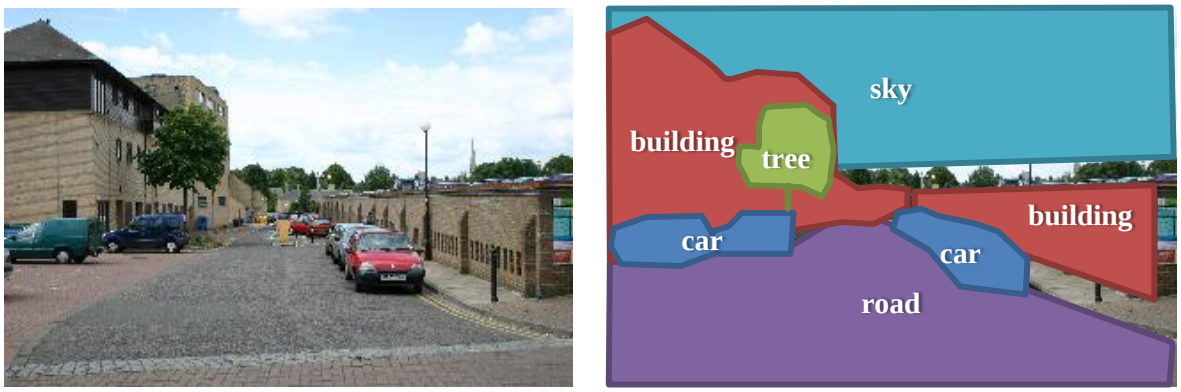


Figure 1.1: *An Example Semantic Segmentation*. A semantic segmentation of an image is one which groups the pixels together by common semantic meaning. Shown is one such segmentation of an image, using as pixel labels the objects in the scene. This information can be used for automatic image annotation, or for compositional search based upon semantics.

useful in general, and thus you may end up with a website where a human has compiled a list of their favorite 50's horror movies, or a page about a book on great horror movies from that era. The web has been this way since its inception, and so we have perhaps forgotten that what we really want to do is to learn more about great horror movies of the 1950s, not click through 50 different websites to compile that information for ourselves. This is the essence of what the semantic web is meant to overcome. Instead of retrieving individual pages which may or may not have the information you want, a truly semantic web would understand the actual meaning of the conversational query you have given it (requiring a working solution to natural language processing) and would be able to assemble a custom-built website which gives you all of the information you could ever want to know about your query, arranged from multiple sources whose content it truly understands (arguably requiring a working solution to general AI). It is no surprise that some insist that a truly semantic web is out of reach.

That being said, a significant effort is currently being made through the use of a new generation of languages, schema and rule definition to enable human content creators to tag data effectively so that intelligent agents can easily learn from it [30]. One great barrier to a

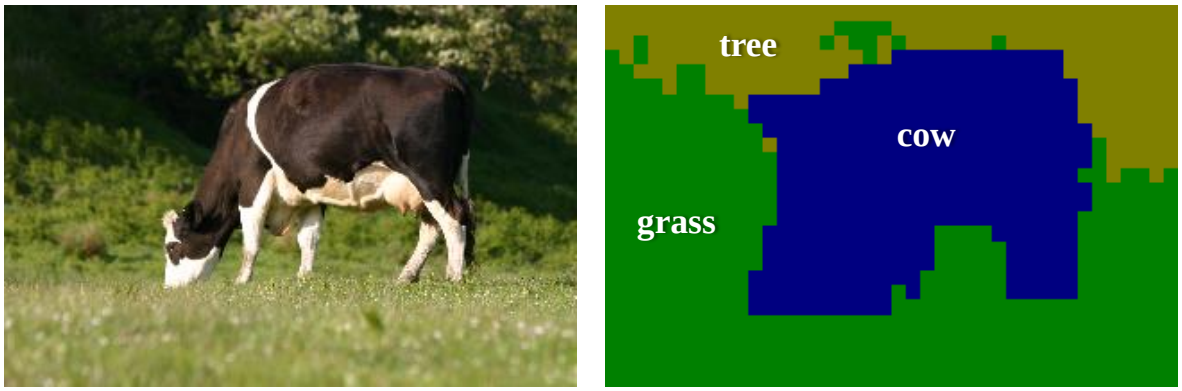


Figure 1.2: *Sample Segmentation Performance*. This is an example segmentation performed by our automatic segmentation algorithm, presented in Chapter 4. Each image grid cell is given a label from a fixed dictionary of categories, chosen as the maximum *a posteriori* category based upon the pixel appearance. This inference is performed by a semantic texton forest, which we introduce in Chapter 3.

semantic web remains, however: the understanding of visual content. It is estimated that, in the US alone, well over 8 billion digital images were shared online by individual users in 2007 [75]. True semantic understanding of an image is represented by a *semantic segmentation* of an image, such as that shown in Figure 1.1. Each meaningful pixel is given a semantic label which describes the category of object which produced it in the scene. There are multiple semantic segmentations of an image, depending on the vocabulary of objects used, but each one gives an understanding of the image's underlying meaning. While this can be done by human annotators [85], it is laborious and the sheer quantity of images involved makes this prohibitive. Therefore, the development of novel methods for the automatic semantic segmentation of images is essential to moving forward.

1.1 Semantic Segmentation

We present a system for performing semantic segmentation in Chapter 4 which uses a novel discriminative machine learning approach, the *Semantic Texton Forest*, which we introduce in Chapter 3. Each forest is a combination of decision trees, and each tree is trained indepen-

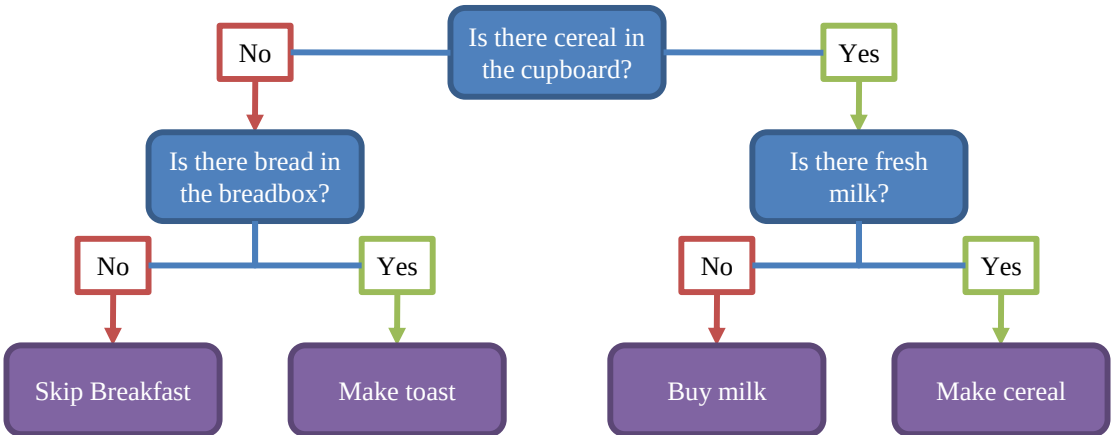


Figure 1.3: *Example Decision Tree.* This is an example decision tree for determining what to do about breakfast. At each node a simple test is performed, and the result of that test is used to determine which child to choose. This process is repeated until a leaf node is reached, with each leaf encoding a particular decision to be made that is based upon all of the tests performed to reach that node.

dently to infer the category label for an image pixel. A decision tree is a directed graph, in which each node has one parent and any number of children (with the root of the tree being the only node without a parent). When making a decision, the system asks a simple question at each node (starting with the root node) to determine which of its children to choose, and repeats the process until a leaf node is reached, where a leaf node is a node which has no children and contains the decision to be made. An example decision tree can be seen in Figure 1.3, in which a decision can be made about what to do for breakfast based upon simple observations. In the case of our system, the decision to be made is the category to which a pixel belongs, and the tests are simple combinations of pixel values. The trees learn directly from the data which combinations of a pixel's nearby neighbors should be used to determine its category. Because it is a tree-based structure, this decision can be made in $O(\log n)$ time. Each tree is trained on a different subset of the data and in a randomized nature, so that each one has a different understanding. By combining together the decisions made by multiple trees, we are able to achieve accurate inference of the pixel labels. An example of our segmentation performance can be seen in Figure 1.2.

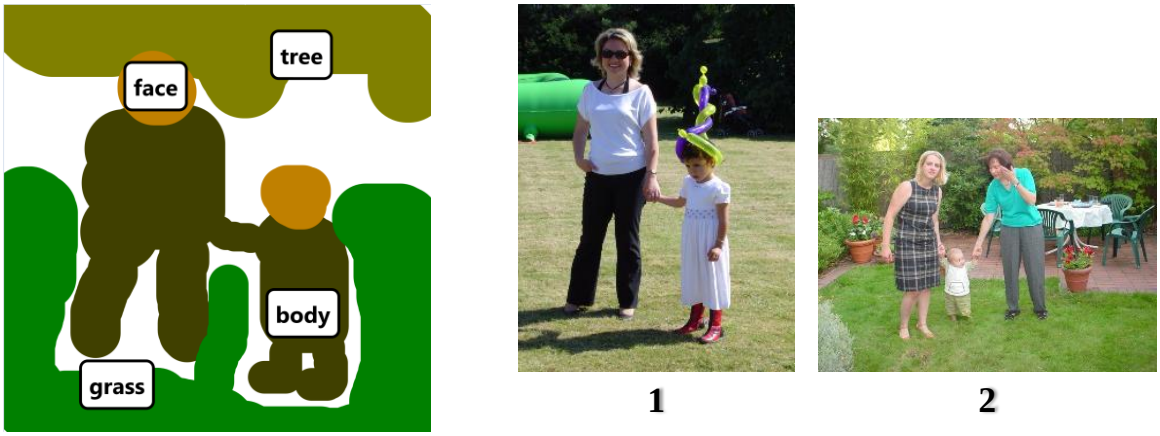


Figure 1.4: *Sample Semantic Composition Query*. The query on the left and its top two results shown are actual results from our semantic image search program, presented in Chapter 7. The user uses a painting interface to demarcate the areas of a target image which are produced by a particular category, resulting in the query image on the left. The system uses a model of their query to infer the likelihood of each image in the database being the target image which produced their query, and uses this likelihood to rank the retrieval results.

1.2 Semantic Image Search

The problem of *query interpretation* centers around how to translate the ideal, desired image in a user's mind into a form which a computer can understand. Our approach is built on the idea that the most essential element of that ideal image is its *semantic composition*. We formulate semantic composition as a Bayesian framework which is capable of representing both the objects present in the user's query and their arrangement in the 2D image plane, and comparing them to known images to evaluate the most likely match. We have developed a paradigm for specifying these constraints consisting of a intuitive, familiar yet very powerful interface: a canvas on which the user can paint a sketch of the image, where each color corresponds to an object category. While this interface provides a method by which to specify an incredible variety of constraints, it also itself constrains the image database search task: the system must be able to search semantically (by object type, *e.g.* cat, bicycle) and by composition (by location in the image, *e.g.* upper left corner, center). An example of a query and its results is shown in Figure 1.4.

1.3 Overview

We begin by examining the state of the art in image categorization in Chapter 2. Specifically, we examine how the bag-of-words model for information can be used to describe and compare images through the use of interest points and descriptors, and demonstrate that by combining complimentary cues together we can improve categorization performance. Building upon this finding, we develop the semantic texton forest in Chapter 3, which eschews interest points and descriptors in favor of learning the image representation directly from the data, and compare it to our categorization results using standard techniques. We then explore its uses for automatic semantic segmentation in Chapter 4 and for image annotation in Chapter 5.

We then turn to applications of this new technique in the rest of the thesis. We introduce the concept of *semantic composition* in Chapter 6, and demonstrate its uses for query representation. The final result is the Palette Search system presented in Chapter 7, which is the first system to allow the user to perform Query by Semantic Composition, in which he specifies not only what he wants in the image, but where in the image it should be present. In our conclusion in Chapter 8, we discuss the limitations of this work and directions for future research. In Appendix A, we discuss previous methods for representing visual data, specifically as they touch upon our methods. Finally, Appendix B documents the challenging datasets which we use in this thesis.

CHAPTER 2

THE BAG-OF-WORDS HISTOGRAM

One of the most important tasks in computer vision is *image categorization*. Categorizing an image consists of determining those categories (*e.g.* forest images, office images, moon images) to which an image belongs. An image can belong to multiple categories, and these categories can be very broad (*e.g.* sports, medicine) or narrow (*e.g.* football, kidneys). Image categorization is one way in which we can perform image retrieval (*i.e.* by providing semantic categories and keywords for an image) and can be used to inform other tasks, such as segmentation or detection.

We begin with image categorization for several reasons. First, it provides an opportunity to examine the state of the art in the semantic understanding of images. Query by Semantic Example image search systems commonly depend on image categorization of some kind to perform their queries [101], and automatic categorization can allow the use of standard text-based retrieval systems [98], which in many ways are more mature than their counterparts in image retrieval. Secondly, it provides a mechanism by which we can introduce various conventions and techniques in computer vision which will be used extensively in the rest of the thesis.

An image cue can be thought of as an information type in the image. Each cue represents the data in the image, but in a different and sometimes orthogonal way from the others. In this chapter we take four common image cues and develop a way of integrating them into

the same model of information representation: the *bag-of-words histogram*[1]. We then examine how combining these cues results in an improvement in image categorization performance.

Combining cues has resulting in significant improvements in image categorization performance. By using per-category cue weights during the training process, a system can learn which cues are best for a particular category as is done by Varma and Ray through the optimization and weighting of SVM kernels during training [99]. An alternative is to learn these weights through cross-validation, as in the work of Bosch *et al.* [10]. In this chapter, we concentrate on separating the cues so that we can record the performance for image categorization using the cues in isolation from each other, and then measure the effect of combining them upon this performance.

2.1 Exploiting Image Information

A color image is rich with information, much like a novel is rich with letters. The various ways of summarizing an image are no more a complete representation of the image than the various ways of summarizing a novel. Can you retell the story of a novel from a word histogram, or relate its main themes from a list of the main characters? It is similarly unrealistic to expect machine learning algorithms to solve increasingly harder tasks in computer vision when they are not given all of the information possible. As shown in Figure 2.1 there are many different kinds of information present in an image. Using all of this information to perform categorization has been a goal of many techniques past and present. Efforts in this area can be grouped into two categories: those which segment the image in some way and those which look at global histograms of some kind.

A characteristic method which utilizes segmentation would be the excellent Blobworld

[1] A full discussion of common cues can be found in Appendix A

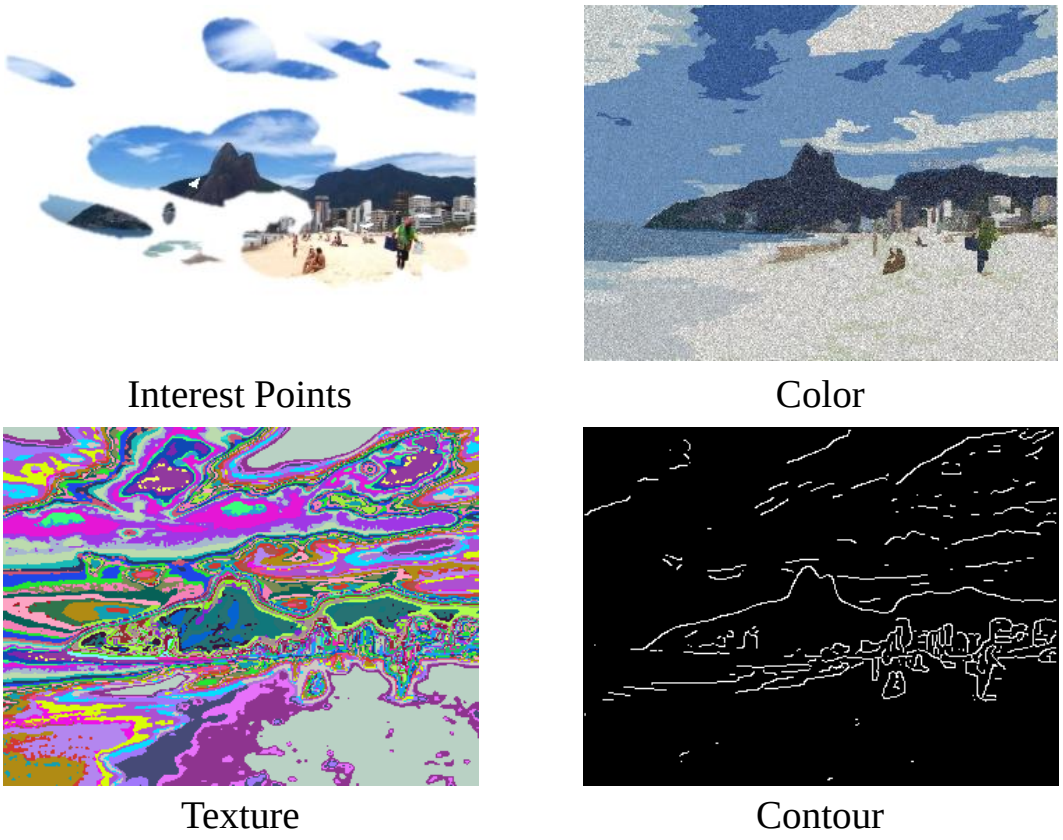


Figure 2.1: *Different Image Cues*. Here is the same image of a beach showing four different image cues: interest points with local descriptors, color, texture and contour. The interest points, computed as in [70], are represented as ellipses, indicating the scale and orientation of the interest point. The descriptor is computed using a warped image patch, where the warp is performed such that the ellipse becomes a circle. Color is represented by regions of an image obtained from a photometric segmentation algorithm [31] which are then assigned to color clusters, consisting of a mean and variance in CIELab color space [50]. The pixels are sampled from this Gaussian distribution for each region. Texture is represented as textons as in [92], with the pixel color corresponding to a texton index. Finally, contour is shown as connected edges.

work [17] and all the research that has grown from it, in particular the methods which treat blobs as visual concepts, much like words [2]. The eponymous "blobs" in this case are not circle-like shapes in the image (as described in Appendix A) but rather small to middle size segments of an image which have a consistent color and texture. Each blob is represented by a feature vector which consists of various elements such as Gaussian filter responses and mean color values, which represent texture, color and to some extent contour for the region. Their experiments concentrated on the Corel data set, which has large numbers of professional photos with four or five descriptive keywords associated with each, with the goal of learning the connections between certain kinds of blobs and the associated image keywords.

Ren *et al*. use segmentation into small, consistent regions called superpixels [81] which are then assigned to one class or another to segment the image (and recognize objects within it). These also recognize the importance of texture, contour and color to a certain degree, but the methods in which they are used seek simply to create large regions which have consistent parts, and thus do not learn anything in particular about how the combinations of these cues denote the category assignment.

Global histogram methods take various forms, from work using the bag-of-words approach [22] to texton histograms [106]. Most histogram-based approaches look at a particular channel of information to exclusion, such as feature points and descriptors [29; 79; 93] or filter bank responses in [77]. There are several approaches which combine channel information using statistical modeling, but again in a segmentation context in which creating regions of consistent explanation is the goal [92; 41].

2.2 Bag-of-Words Histograms

The bag-of-words histogram as a tool for information representation originates in the text categorization community [21; 64; 52], where it was used to describe documents by the frequency of certain words within them. Every word has an inherent set of topics where it is used more often than others. By looking at which words correspond to which topics, a system is trained to look at a histogram of word occurrences within an unseen document and assign it a topic. For example, if a new document has many instances of the words "bank", "money", and "transaction", then it is assigned to the topic "financial".

The system was first used in computer vision by Sivic *et al.* [94], with work by Csurka *et al.* [22] showing its applicability to object recognition. The key innovation lies in freeing up the concept of a "word" from being a collection of letters. Instead of a text dictionary, we build a feature dictionary D which is based upon the set of all the features in our training images. We describe the method by which we build this dictionary in Section 2.3. An image I is described by a set of features F_I by way of a feature extraction algorithm $F()$ ($F_I = F(I)$). $F()$ takes as its input an image and outputs a set of feature vectors. Each feature $f \in F_I$ is then assigned to the "word" $d_i \in D$ which it most resembles using a lookup function $L()$, thus allowing a bag-of-words histogram H to be built from the image:

$$H[i] = \sum_{f \in F_I} \begin{cases} 1, L(f) = d_i \\ 0, \text{otherwise} \end{cases} \tag{2.1}$$

where $|H| = |D|$, both are indexed by i ($d_i = D[i]$) and $|H|$ is used to denote the length of vector H. Thus, in order for a cue to be described using a bag-of-words histogram, we must designate a feature extraction algorithm $F()$ for that cue, and find a lookup function $L()$ to which it is best suited. In our experiments $F()$ changes from cue to cue, but we have chosen

$F()$ so that $L()$ is always calculated using the L2-norm distance, specifically

$$E(f,d) = \left(\sum_{j}^{M} (f[j] - d[j])^2 \right)^{\frac{1}{2}} \tag{2.2}$$

$$L(f) = \text{argmin}_d(E(f,d)) \tag{2.3}$$

where j indexes the feature vector and $M = |f| = |d|$. M is dependent upon $F()$, and can be different for each cue.

The problem with using a bag-of-words histogram to represent the information contained in an image cue is that, as noted by Fergus *et al.* [32] and others [79], visual words inherit the problem of ambiguity from their textual counterparts. Meaning ambiguity comes in two forms: polysemy or synonymy. A polysemous word is one which has multiple definitions. Two synonyms are words which have the same meaning. These two forms plague textual analysis, with the entire field of word sense disambiguation devoted to determining which sense to assign to a particular occurrence of a word. In image space, visual words chosen are by their nature very general and are often found in many different categories. By using different cues together, each cue can use the others as context, thus aiding in disambiguating the meaning of the visual words. The knowledge that an image has many blue patches on its own is not very useful (as many things are blue). However, in combination with the knowledge that it also has quite a lot of smooth texture, a long, straight horizontal line, quite a bit of green and some grassy texture it begins to look very much like those blue patches indicate sky, and that the image depicts an open country scene.

The system which we will use to explore the bag-of-words histogram as a tool for image categorization in this chapter is shown in Figure 2.2. We will first explore ways of incorporating interest points and descriptors, color, texture and contour into this system. These are not the only ways to do so, by any means, but they provide a unified model of representing

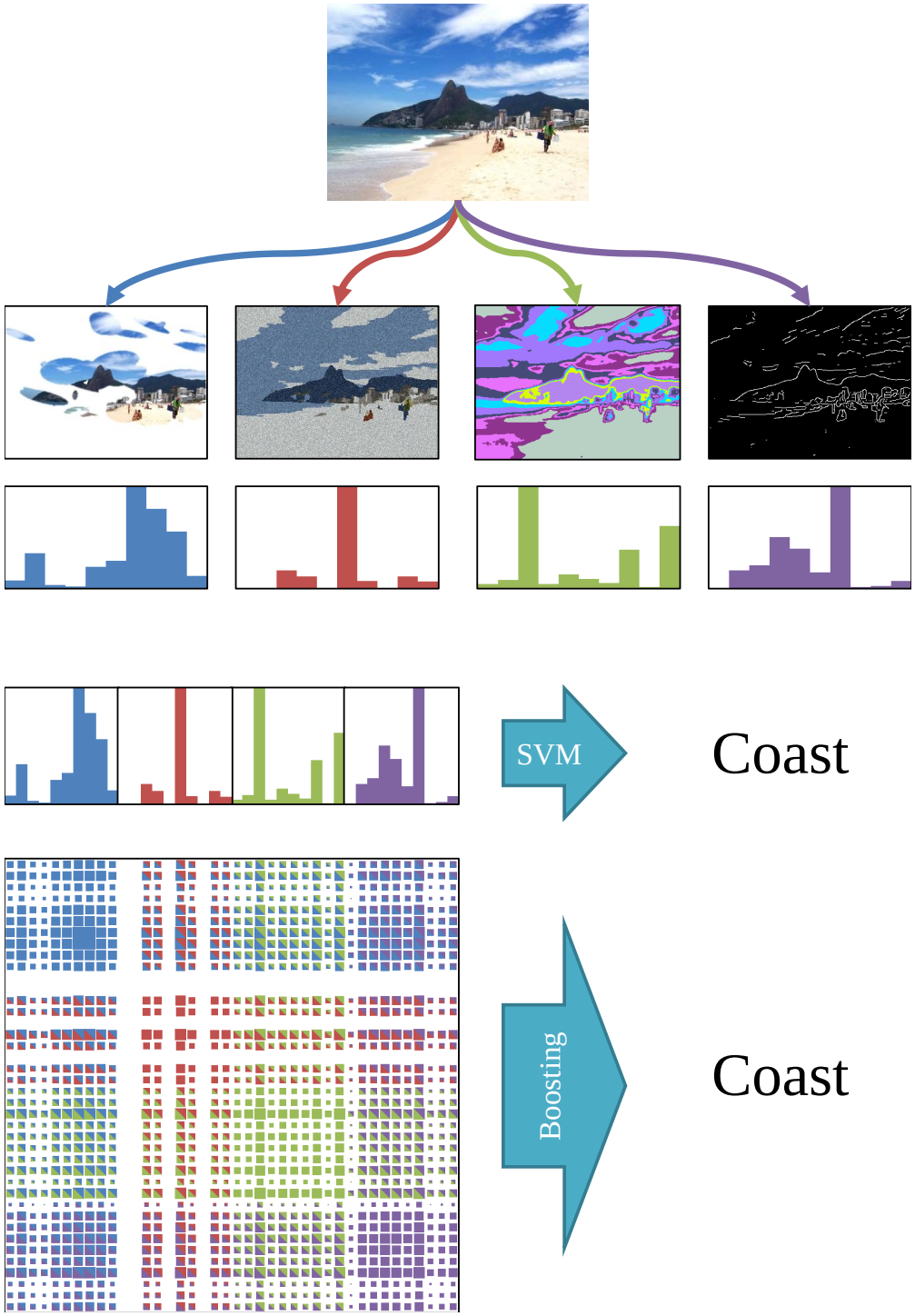


Figure 2.2: *Cross-Cue Categorization*. An image is decomposed into four image cues. Each of these cues is described using a bag-of-words histogram, formed by a cue-specific vector quantization function $F()$. For use in Support Vector Machines, they are joined together (Section 2.3.1). For Joint Boosting, a co-occurrence matrix is formed between appended histograms and each individual cell of the matrix treated as a weak learner (Section 2.3.2). The result of using the cues together is improved categorization performance (Section 2.4).

Figure 2.3: *Interest Points and Descriptors*. We use the Harris-Hessian interest point detector of Mikolajczyk *et al*. [70] and Lowe's SIFT descriptor [65] as $F()$ for this cue. In this image, the ellipses produced as a result of the interest point detection on the image on the left are shown on their own on the right. The ellipse indicates the scale and orientation of each point. The descriptor is computed on the warped image patch, where the warp is governed by the ellipse, namely that under the warp the ellipse is a circle. This is done to add a degree of invariance to affine transformations.

these cues in order that we may compare them and see how they can combine to improve performance.

2.2.1 Interest Points and Descriptors

While each pixel in an image is potentially interesting, in practice there are several specific points in the image which are useful for computer vision. These can be divided into three groups: corners, edges and blobs. The advantages that these points have is that they can be detected reliably in different images of the same scene because they are created by objects within that scene. The difficulty comes in finding the correspondence between two images of the same scene, which is why interest points are often coupled with local descriptors, which describe the area around them. For a full discussion of interest points and descriptors, see Appendix A.

When binned into a bag-of-words histogram, these local points and descriptors can be thought of as cataloguing the various basic parts in an image. The words in the dictionary tend to represent basic visual building blocks of the various categories. For example, if

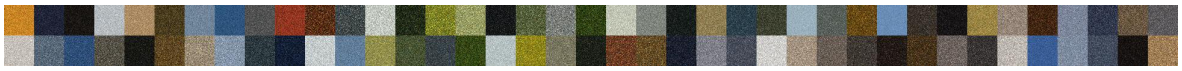
the training data is a collection of face images then the words will be things like eyes and noses. If the training data is a set of bicycle images, then likely words are wheels and gears. They can also represent basic image structure like circles, bars and edges and as such these histograms can also encode the generalized textures that occur around interest points. An example of an image and the interest points and descriptors extracted can be seen in Figure 2.3.

The bag-of-words histogram representation has been used extensively with interest points and descriptors, with the first efforts being those of Sivic *et al*. [94] and Csurka *et al*. [22] and then followed by many others [29; 79; 93; 84; 110]. These techniques have used various combinations of detector and descriptor, with the most common combination being some Laplacian-based blob detector (such as the Difference of Gaussians) and Lowe's SIFT descriptor [65]. We use the Harris-Hessian detector of Mikolajczyk *et al*. [70] with the SIFT descriptor as our feature extraction algorithm $F()$ for this cue.

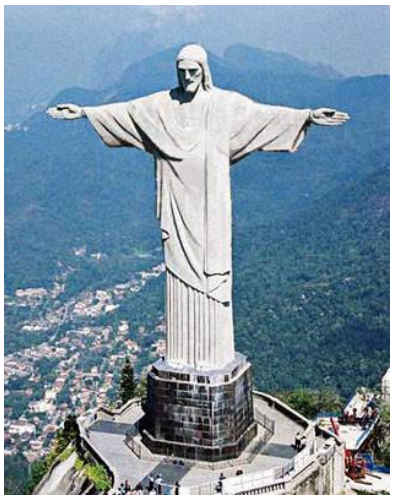
2.2.2 Color

Color is an important and rich cue for an image, but one which is also difficult to handle well. All color analysis is plagued by lighting problems, in which the same scene under different lighting and camera conditions can result in vastly different coloring. This, combined with the difficulty of finding an appropriate color space and the non-Euclidean nature of these spaces, makes creating effective color descriptors a non-trivial endeavor.

There is much to say on the problem of color constancy and its relation to image categorization (see [3] for a good discussion) which is outside of the scope of this thesis. We include a brief discussion of various color spaces in our treatment of color in Appendix A. For our color descriptors, we use the mean and standard deviation of the L*, a*, and b* values for a segment of the image. These values are those updated from Hunter's original

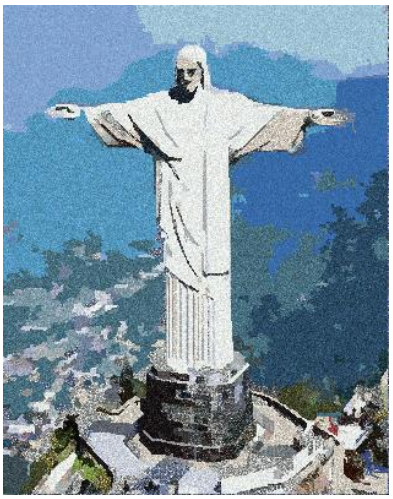


Color Dictionary



Original Image

Color Segmentation



Vector Quantization

Figure 2.4: *Color Clusters*. An image is segmented into regions of consistent color, and each region is represented by their mean and standard deviation in CIELab space. These are then vector quantized using a code-book, such as the one displayed. The vector-quantized regions and codebook are represented by sampling from the Gaussian function (to give a visual sense of their variance).

Lab color space [49; 48] by the Commission Internationale d'Eclairage (International Commission on Illumination) (see Section A.1.3 for details). This is inspired by the treatment of color by Barnard *et al*. [2], but where they use Normalized Cuts to segment an image we use the graph-based algorithm of Feltzenszwalb and Huttenlocher [31]. The reason the mean and standard deviation are used is that, in addition to giving a sense of how textured a region is (with a higher standard deviation equating to a high level of color variance and hence, texture) it also allows regions which suffer from color distortion from camera and scene conditions to produce similar descriptors.

We oversegment the image to create a large number of small, consistent regions, as can be seen in 2.4 and compute the mean and standard deviation of L*, a*, and b* for each. For future reference, we will call this simple descriptor the Color Cluster, and this process forms the feature extraction algorithm $F()$ for this cue.

2.2.3 Texture

One of the most important image cues is texture, whether it be the sand on a beach, a smooth sky, or repeating blades of grass. The concept of the texton was introduced by Julesz in [56] as the basic building block of texture. In much the same way that many different colors can be mixed from a collection of primary colors, texture can be built from the mixing in proportion of various textons. The specific definition of a texton described by Leung and Malik in [63] is the one which has seen the most use, and is described in Section A.3.2. The simple filters which they use approximate Gabor filters, which were proposed for texture description by Daugman in [23], and are motivated by mammalian visual systems [66]. Examples of these filters can be seen in Figure 2.5.

When binned into a bag-of-words histogram, textons can be thought of as describing the kinds of texture building blocks in an image. For example, in an image of grass and sky

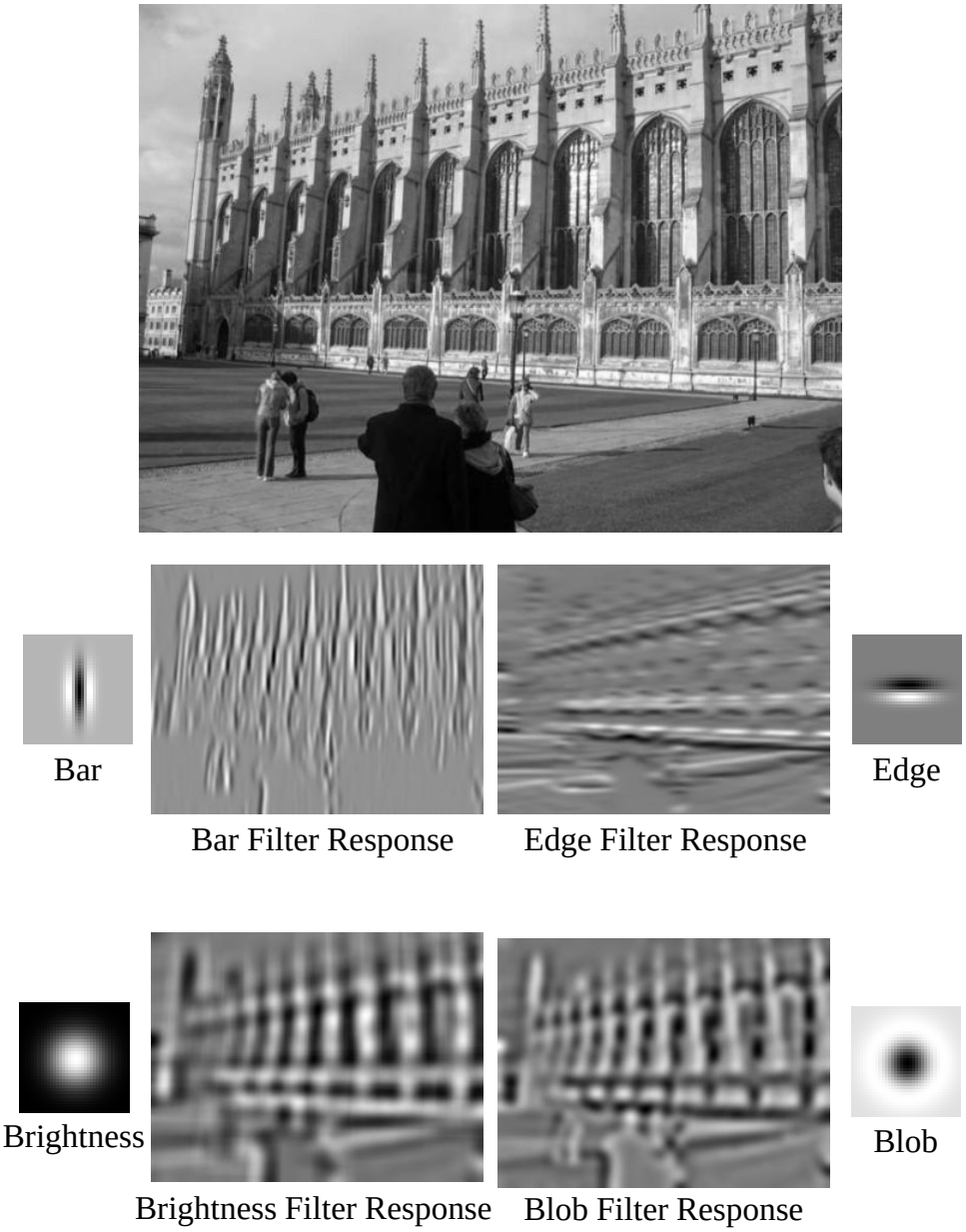


Figure 2.5: *Simple Cells*. The basic kinds of simple cell, directed and undirected filters. The Bar filter is the second derivative of a Gaussian in one direction elongated by a Gaussian perpendicular to it. The Edge filter is the first derivative of a Gaussian, similarly elongated. The Brightness filter is bivariate Gaussian with diagonal covariance, and the Blob filter is a Laplacian of a Gaussian.

there will be many bar-like textons (formed from firings of the second derivative bar filter) and many smooth textons (formed from firings of the Gaussian filter). Similarly, city scenes will have many edge textons from the straight lines of roads, windows, doors and signs. The feature extraction algorithm $F()$ for texture consists of a subset of the full feature bank described in Appendix A, that being 11 filter responses extracted at each pixel, which form the descriptor. These filters consist of

- Three Gaussians at $\sigma = \{1, 2, 4\}$
- Four Laplacians at $\sigma = \{1, 2, 4, 8\}$
- Two horizontal Gaussian derivatives at $\sigma = \{2, 4\}$
- Two vertical Gaussian derivatives at $\sigma = \{2, 4\}$

which are convolved on a patch of size $(6\sigma + 1) \times (6\sigma + 1)$ centered on the pixel.

2.2.4 Contour

The edges in an image have been used in Computer Vision as an important cue for much of its history due to the simplicity with which they can be extracted, the level of information reduction they provide and the invariance they show to many image conditions. An edge in an image is defined as a point where there is a large intensity gradient, with the magnitude of that gradient giving an edge its strength and the direction of the gradient giving the edge an orientation. By stringing edges together using an edge detection algorithm like that proposed by Canny [15] it is possible to build up continuous edges in the image, or contours.

The problem with contours is that they are by nature a continuous entity. Since we wanted to create a bag-of-words histogram which describes the contour in an image, we needed to fine a way of making them discrete; in other words we need a way to localize a contour, and a way to describe that contour with a feature descriptor.

For the purpose of localizing contours, we use the individual edge points found using Canny Edge detection, but instead of only using edges at a single scale of the image, we instead extract edges at multiple levels of an image pyramid. The reason for extracting edges at multiple levels is to provide an invariance to scale for our descriptor. We build the pyramid by having the original image as the base, and then convolving using a Gaussian with a standard deviation of σ_c (for the purpose of our experiments we used $\sigma_c = \sqrt{2}$) and performing Canny edge detection. The base octave is then convolved using a Gaussian of $\sigma_m = \sqrt{\frac{3}{N}}\sigma_c$. This is repeated N times (we used $N = 1$) to result in an image equivalent of one which was convolved with a Gaussian of $2\sigma_c$, thus allowing us to safely downsample the image by 2 to provide the next level. This is repeated until the dimensions of the image go below a certain level (we used a width or height of 32 pixels).

Once we have extracted edges from the image, we have to find a way of describing the contour in the neighborhood of an edge pixel. One of the most effective contour descriptors is the shape context [6]. It uses a log polar histogram to describe the location of all edges in relation to a particular edge. We use it as our main inspiration for Discrete Contours. We implement the log polar histogram using a grid, computed as shown in Figure 2.6, which allows for much greater efficiency and flexibility (by shifting the focus to the local contour fragment as opposed to all edges in an image). The contributions to each bin are weighted by the edge magnitude (as done in [70]) and distributed by orientation. Each outer ring bin is subdivided into 8 equally-spaced orientation bins, the middle ring into 4 bins, and no subdivision in the inner ring. Thus, for each pixel in a bin, the magnitude of the edge at that pixel is interpolated into two orientation bins depending on the edge orientation (again, excepting the inner ring). This results in a 104-dimensional descriptor which describes a contour fragment very well, which we have named a Discrete Contour.

Our feature extraction algorithm $F()$ for contours uses this method with the settings de-

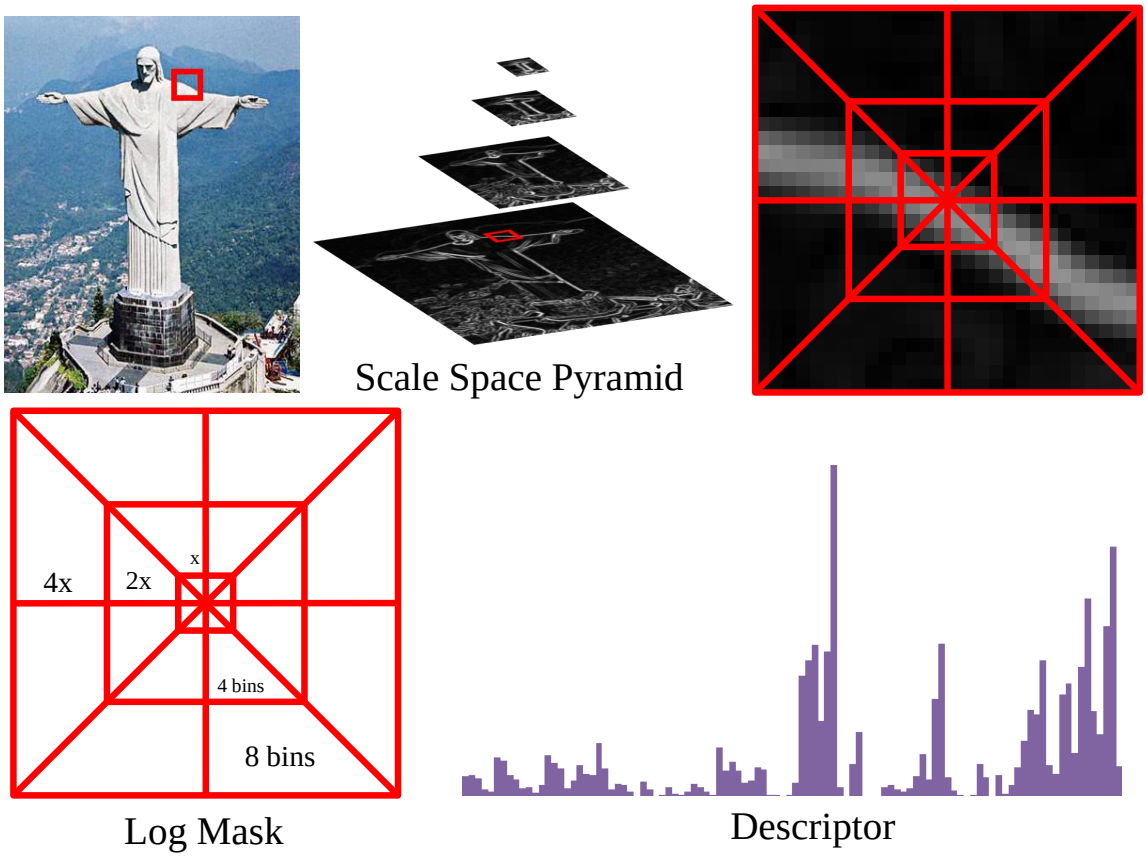


Figure 2.6: *Discrete Contour*. This is a representation of how the discrete contour descriptor is computed. The log mask on the bottom left is centered at a Canny edge at a particular scale and the magnitudes of the edges at each pixel added into the appropriate orientation bin for the grid. The outer ring of grid cells have 8 orientation bins, the middle ring 4, and just 1 in the inner ring. In our experiments we used x = 2 pixels (making the log mask 28 pixels square). Each pixel's magnitude is interpolated between the two nearest orientation bins, and in the case of pixels lying on the borders between grid cells, the magnitude is split equally between the two grid cells.

scribed in the text. Instead of using every edge location returned from the Canny algorithm, we densely sample all of the edges (as was the case in the original shape context work) [6].

2.3 Image Categorization

In this thesis, we will concentrate on performance using two different datasets, the full details of which can be found in Appendix B, including sample images, numbers of images and other pertinent information. In this chapter, we will be looking at performance on the Scenes database. It was first used by Oliva and Torralba in [76], has 8 categories (coast, forest, highway, inside city, mountain, open country, street and tall building) and we used 100 training images per category as per [76].

For our experiments we created 10 different train/test splits. In each split, the training images were randomly sampled from the category domain and the test images consisted of the remainder. For each, the feature extraction algorithm $F()$ for each image cue was used to extract descriptors from all images. For each image cue, we then sampled 5000 descriptors from the domain of training descriptors for each category. Following the technique used by Zhang *et al*. in [110], we clustered these 5000 descriptors separately for each category using K-medoids clustering (K = 10) and appended the cluster centers into a dictionary for the cue. Using these dictionaries, we then vector quantized the descriptors from all images to compute a bag-of-words histogram for each image cue for each image. It is important that this is done once for each split, as the dictionaries depend on the training images and as such the bag-of-words histograms will change slightly depending on which collection of training images is chosen.

From this point, the process is somewhat dependent on the learning method used, but the task for each learner to accomplish is to categorize each test image as belonging to one of the categories in the dataset. All test images were drawn from the categories in the database,

and so the situation presented to each learner is always to choose one category versus all the others. For each learner and dataset we performed 5 sets of 10 experiments: one each for each cue in which the learner was trained on only the histograms from that cue, and one where it was trained using a combination of all the cues. Both learning methods produce as their output for an image a probability distribution, $P(X_I = x)$ (where $\{x\}$ is the domain of the category random variable X and X_I is the variable for an image I) over all categories. For labeling, we define A, the labeling function, as $A(I) = \text{argmax}_x P(X_I = x)$.

2.3.1 Support Vector Machines

Our two learning methods use very different forms of sparsity. Support vector machines [87] choose a sparse set of example images which delineate the boundaries between categories. For our experiments we used the multi-class probabilistic SVM from the libSVM library [18]. The training data consisted of the training images from each category in equal proportion so as not to bias the SVM towards a particular category. The particular SVM used was a multiclass C-SVC with a radial basis function kernel. The training data was scaled to have a range from -1 to 1 for each vector index, and the scaling transform then used on the test data. We performed 5-fold cross-validation on the training data to select C and γ for the model. To combine the various cues, we appended the histograms together. The inherent problems in range of different cues (as different cues have a different total number of features per image and as such a different histogram magnitude) were dealt with by way of the aforementioned index-based scaling.

2.3.2 Joint Boosting

The second of our learning methods, Joint Boosting, uses a sparse set of features in weighted combination to make its decisions. We used the implementation from [92] in our experi-

ments. This system uses weak learners in which a particular feature is used to divide the data by way of a threshold. By combining these weak learners together it is able to divide the data into various categories. Since this method requires a large, sparse set of features we utilized the co-occurrence matrix of our histograms. Thus, for a histogram H, a co-occurrence matrix Z is constructed as follows

$$Z[i,j] = \begin{cases} \frac{H[i]\times(H[i]-1)}{2}, i = j \\ H[i] \times H[j], \text{otherwise} \end{cases} \tag{2.4}$$

Constructing a co-occurrence matrix in this way has a variety of benefits. First, it allows for an inherent context to be built into the features being learned, as instead of simply learning the presence of red, the system is instead learning the presence of red and blue together. More importantly, in the case in which H is an appended histogram of all the cue histograms, the system is learning the presence of blue and sand texture (implying ocean), or of straight lines and repeated rectangle textures (implying buildings).

The combined case causes an inherent scale-based dilemma, as different cues have different magnitudes of detection, potentially allowing one cue to overpower the others and create a bias. To deal with this we perform two operations. The first is to apply term frequency/inverse document frequency analysis [86] when creating cue histograms for boosting. This technique reduces the influence of common terms while scaling a term histogram so that it is unbiased by length of a document, and as such is perfect for our uses. In this method, the original histogram, after being calculated as in Equation 2.1, is transformed as follows

$$\phi[i] = \frac{H[i]}{\sum_i H[i]} \tag{2.5}$$

$$H_{\phi\varphi}[i] = \phi[i]\varphi[i] \tag{2.6}$$

The inverse document frequency (φ in the previous equation) is calculated from the training image set P as follows

$$\varphi[i] = \log \frac{|P|}{|\{I : H_I[i] > 0\}|} \tag{2.7}$$

This has the effect of scaling the data to the range of 0 to 1. To create meaningful counts for co-occurrence we therefore scale this to the range of $[0, 1000]$. The second operation is performed on the individual co-occurrences. Since the weak learners must sweep through a range of values to find an ideal threshold for dividing the data, it is useful to have each co-occurrence have the same range. Thus, we look at the training data and find a scaling transform which achieves this, which we also apply to the test data. The number of rounds used for boosting was determined by dividing the training data into training and validation sets and finding the average number of rounds before the elbow in the curve of the sum of the diagonal of the confusion matrix versus rounds.

2.3.3 Evaluation Method

For each experiment, we computed a confusion matrix from the results, in which a matrix M is assembled from the test data R using a classifier A and ground truth information G as follows

$$M[i, j] = |\{I : I \in R, G(I) = x_i, A(I) = x_j\}| \tag{2.8}$$

where i and j index X. Thus, the diagonals of this matrix indicate how many correct labels were given by A (the labeling accuracy α_{x_i}), which we report as the proportion

$$\alpha_{x_i} = \frac{M[i, i]}{\sum_j M[i, j]} \, . \tag{2.9}$$

2.4 Results

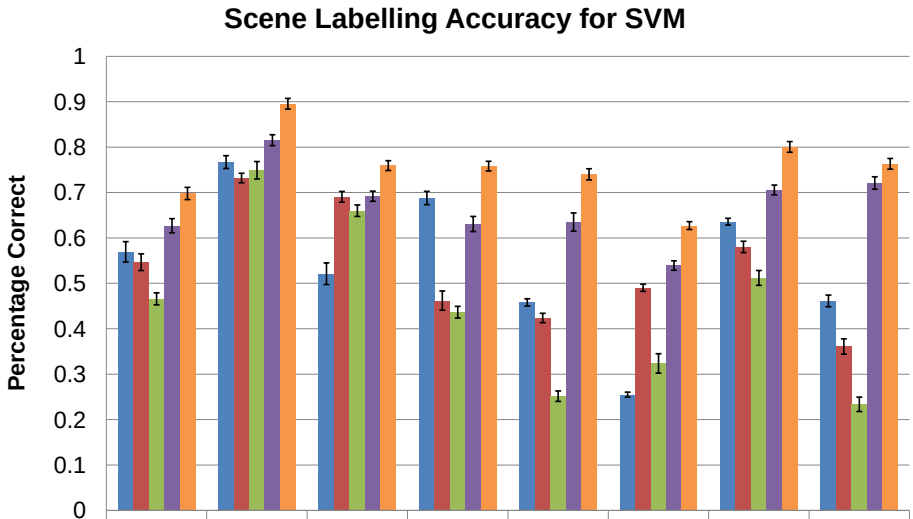
Our results demonstrate that using multiple image cues results in better categorization performance than using those cues separately. They also show that by using a very straightforward, simple and unified framework for representing information in those cues we can achieve equivalent to or better than state-of-the-art performance on both data sets with both learning methods without any tailoring to the data.

In Figure 2.7, which displays the labeling accuracy for SVM learning and Joint Boosting learning, it can clearly be seen how the individual cues work well on a subset of categories and the combined cues exceed the performance of the individual cues on all categories. As expected, combining cues not only fills in the weak points of the various cues, but also uses the context provided by additional cues to increase performance in each category.

Due to the way in which the labeling function A works (as described in Section 2.3), the label chosen for an image is that which has the maximum likelihood in the distribution $P(X_I = x)$. Since we are looking at global histograms for the image, this results in errors for images which have representatives of different categories within them. For example, images which lie on the boundary between two categories (*e.g.* open country images with many trees or forest images with sparse foliage).

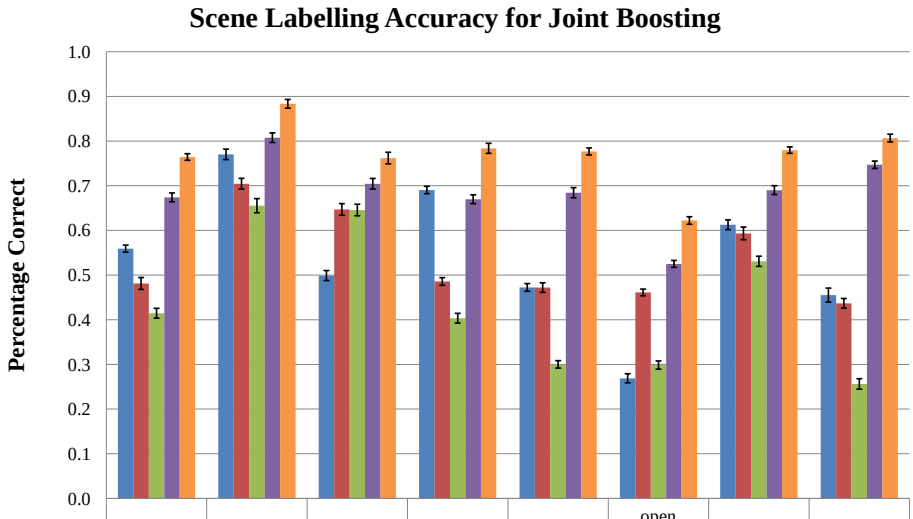
2.5 Summary

In this chapter we have explored the state of the art in image categorization using bag-of-words histograms and machine learning, and shown how combining the different cues in an image into a single framework can result in improved performance regardless of the learning method used. However, this method of combining cues is limited by several factors. First, since the cues are used separately, the system is unable to exploit them together by



	coast	forest	highway	inside city	mountain	open country	street	tall building
Points	0.5694779	0.7671052	0.52125	0.6879807	0.4580292	0.2548701	0.6359375	0.4613281
Color	0.5465863	0.7320175	0.690625	0.4620192	0.4237226	0.4902597	0.5802083	0.3609375
Texture	0.4658634	0.7491228	0.66	0.4365384	0.2514598	0.3237013	0.5119791	0.2335937
Contours	0.6269076	0.8153508	0.691875	0.6307692	0.6350365	0.5392857	0.7057291	0.7210937
Combined	0.6979919	0.8956140	0.759375	0.7581730	0.7401459	0.6272727	0.8005208	0.7632812

(a)



	coast	forest	highway	inside city	mountain	open country	street	tall building
Points	0.559	0.770	0.499	0.690	0.472	0.269	0.613	0.455
Color	0.481	0.704	0.647	0.486	0.472	0.461	0.593	0.437
Texture	0.414	0.655	0.646	0.403	0.300	0.298	0.531	0.256
Contours	0.674	0.807	0.704	0.670	0.684	0.525	0.690	0.747
Combined	0.764	0.883	0.762	0.784	0.777	0.622	0.780	0.807

(b)

Figure 2.7: *Labeling Accuracy*. These graphs display the mean labeling accuracy for each category, measured as the value on the diagonal of a confusion matrix across all ten train/test splits, for all of the cues separately and then combined for both the object and scene data sets when using (a) Support Vector Machines and (b) Joint Boosting. The error bars are calculated as the standard error of the mean across all train/test splits. Note that while some cues work well on one category or another, the combined cues work better on all categories.

finding which combinations are unique to an instance of a category. The co-occurrence matrix used with Joint Boosting attempts this somewhat, but there are better methods of doing so. Similarly, due to the extraction of interest points the system is making a prior assumption about which parts of an image are important. In the next chapter, we will introduce the semantic texton forest, which addresses both of these issues while opening up an avenue by which the segmentation of an image into regions of consistent semantic interpretation is possible.

CHAPTER 3

SEMANTIC TEXTON FORESTS

In Chapter 2 we demonstrated that combining different image cues results in an improvement in categorization performance. The manner in which we combined these features was suboptimal, in that we appended histograms from different bags-of-words vocabularies together instead of learning vocabularies across cues. The co-occurrence matrix approach encoded this to some degree, but there is still the fact that all four cues had different methodologies and had to be computed separately, resulting in a very inefficient and inelegant framework for cue combination. In this chapter we demonstrate that we can achieve the same level of performance for image categorization using a new framework for representing visual data, the *semantic texton forest* (STF). This is an extension of the Random Forests work of Breiman [12], specifically the work of Geurts *et al*. [36], to the problem of semantic segmentation and image description. We will first give a brief background on randomized decision forests in the context of semantic segmentation, and then explore the effects of various training parameters on pixel categorization performance, and end with experiments which show that a semantic texton forest can achieve the same categorization performance as the previous techniques.

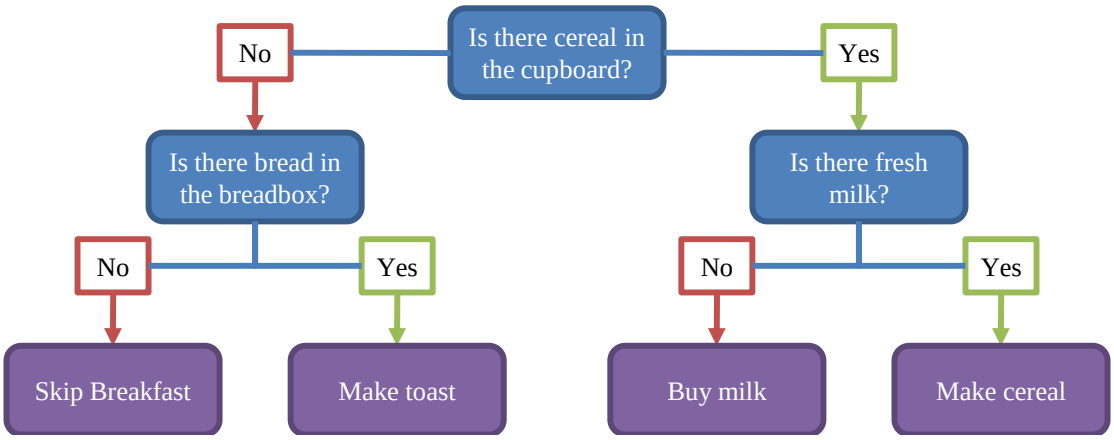
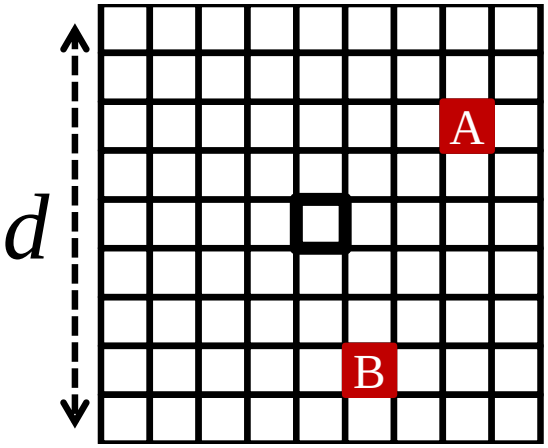


Figure 3.1: *Example Decision Tree.* This is an example decision tree for determining what to do about breakfast. At each node a simple test is performed, and the result of that test is used to determine which child to choose. This process is repeated until a leaf node is reached, with each leaf encoding a particular decision to be made that is based upon all of the tests performed to reach that node.



Label	Test Domain
1	$A[c_0]$
2	$\log(A[c_0])$
3	$A[c_0] + B[c_1]$
4	$A[c_0] - B[c_1]$
5	$\lvert A[c_0] - B[c_1] \rvert$
6	$A[c_0] \log(B[c_1])$
7	$A[c_0] \times B[c_1]$
8	$A[c_0] / B[c_1]$

Figure 3.2: *Test Square.* The questions in a semantic texton forest consist of pixel combinations within a square neighborhood centered on the pixel to be categorized of size $d \times d$. c_0 and c_1 are channels in the image, *e.g.* R,G and B in RGB images. It is not necessary that $c_0 = c_1$. In addition to the tests shown, we also use Rectangle features [92] and Haar-like features [104]

3.1 Randomized Decision Forests

The randomized decision forest is a machine learning technique which can be used to categorize individual pixels of an image [1; 36]. They are based on the concept of a decision tree, which is a construct that is used extensively in data mining [13] and machine learning [12], and which has a wider application in many fields as a planning tool (often in the form of a closely related structure, the influence diagram [46]). In its simplest form, a decision tree consists of a hierarchy of questions which result in a decision, as shown in Figure 3.1. In order to make a decision, one starts at the root, and uses the answer to the question at a node to continue to one of its children, until a leaf node and thus a decision is reached. We want to use decision trees to categorize individual image pixels, and as such the questions at each node are going to be based on image information, specifically mathematical combinations of the each pixel's neighbors as shown in Figure 3.2.

A randomized decision forest is a combination of many different decision trees, each of which has a different set of questions for its nodes, and potentially a different structure from the other trees. The 'randomized' part of the moniker deals with the way in which the trees are trained, in that instead of choosing questions manually or exhaustively from a predordained domain of possible questions, the questions are generated completely at random and chosen according to a performance metric, typically information gain [36]. In our case, the basic unit of categorization is an individual pixel p, and we must determine its category c. Each tree is trained on a subset of the data following the method outlined in Section 3.2.1, using pixel-level ground truth training data such as that in Figure 3.4. This results in a tree with nodes n, and leaf nodes l. Associated with each node is a learned class distribution $P(C_p = c|n)$ where $\{c\}$ is the domain of the category random variable C and C_p is the variable for a pixel p. An example semantic texton tree can be seen in Figure 3.3, in which

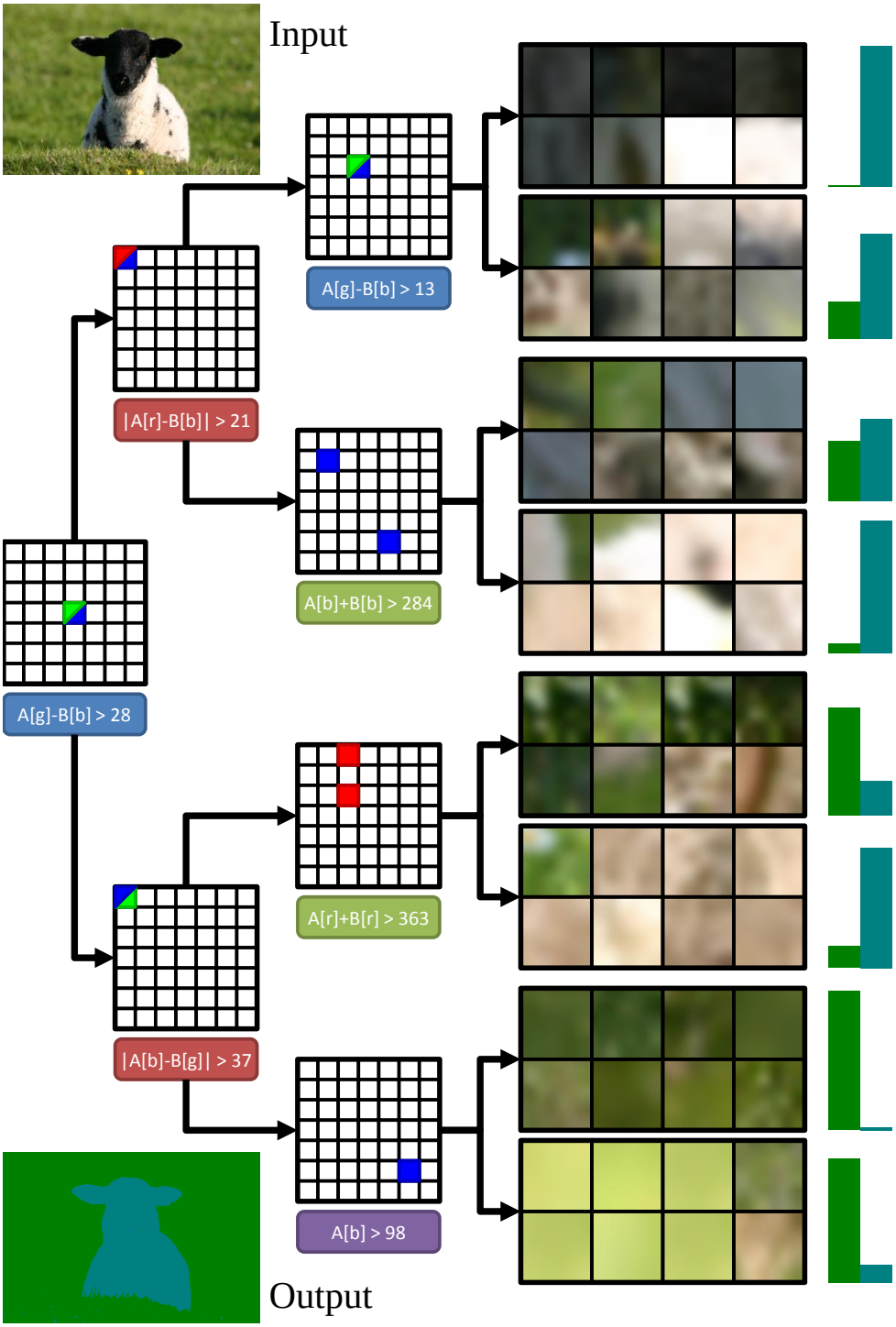


Figure 3.3: *Sample Semantic Texton Tree.* This is an actual semantic texton tree, trained on 23 images of grass and sheep as described in Section 3.2.1. We show the tests performed at each branch node as the test grid ($d = 7$) with the square used for the text filled in with the color corresponding to the RGB channel used. In the case where $A = B$, we split the grid cell and show both colors. The leaf nodes are represented by 8 patches sampled from the training pixels which reached those nodes and the distribution $P(x|c)$ for that leaf node. The green color represents grass, and the blue color represents sheep. The input image is an unseen test image, with the resulting semantic segmentation shown below.

Figure 3.4: *Training Data*. A tree is trained on ground-truth labeled images like these above, in which a semantic label is associated with each pixel of an image.

a tree has been trained on sheep and grass images and can effectively segment an image according to these two semantic categories.

When a new pixel is to be classified, the whole forest achieves an accurate and robust classification by averaging the class distributions over the leaf nodes $L(p) = (l_1, \ldots, l_T)$ reached by the pixel p for all T trees:

$$P(c|L(p)) \propto \sum_{t=1}^{T} P(c|l_t)P(t) \; . \tag{3.1}$$

An example of the overall structure of the forest can be seen in Figure 3.5.

Existing work has shown the power of decision forests as either classifiers [11; 62; 67] or a fast means of clustering descriptors [72]. The trees are fast to learn and evaluate since only a small portion of the tree is traversed for each data point, making them ideal for computing a soft distribution over categories for each pixel. The result of a decision forest which has been applied to an image is either a leaf histogram, *i.e.* a histogram of pixel counts over the leaf nodes, or a tree histogram, a superset of the leaf histogram which includes counts at all of the branches. Naturally, a tree histogram can be computed from a leaf histogram if the tree structure is known.

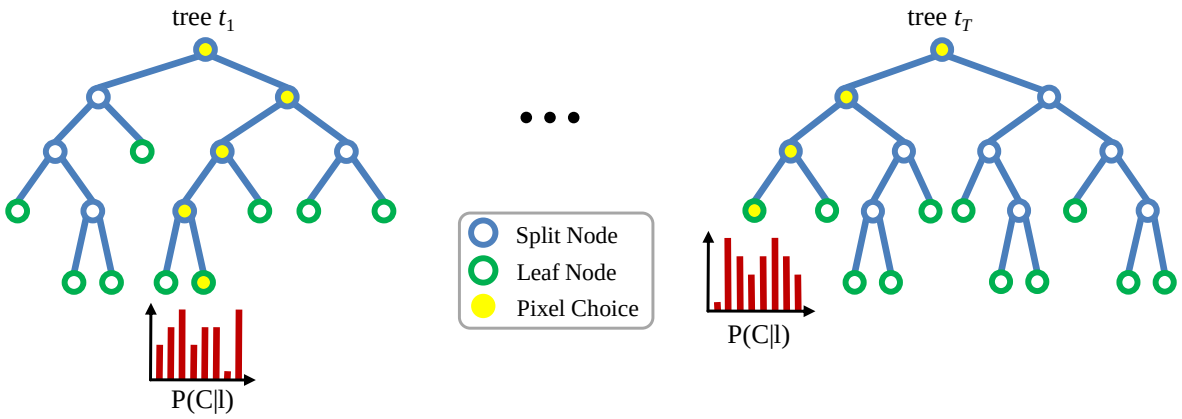


Figure 3.5: *Semantic Texton Forest Structure*. The forest is made up of T binary trees. Each branch node n in a tree (blue circles in this figure) has a test associated with it, and based upon the result of that test one or the other child is chosen. When a leaf node l in a tree t is reached (green circles), the $P(C|l_t)$ distribution for that leaf is used as a soft category decision for the test pixel. In this figure, a sample decision path for each tree is denoted by a series of yellow circles. The final decision is a combination of $P(C|l_t)$ for all $t \in T$.

3.1.1 Supervision

The basic unit of inference for a semantic texton forest is a tuple (p, c), where p is the pixel and c is its category label. Thus, in order to train the forest, labeled image data is required. In the case of *full supervision* this data takes the form shown in Figure 3.4, where each pixel is given a training label. This data can require significant effort to gather and is therefore relatively scarce, but the resulting forest will be able to give more accurate segmentations due to the unambiguous labeling of the pixels.

During training, the distribution $P(C|n)$ is computed as a normalized histogram of the training tuples which reached a particular node n:

$$P(C|n) = \frac{H_n[c]}{\sum_c H_n[c]} \tag{3.2}$$

where $H_n[c]$ is the number of tuples (p, c) which passed through a node n during training. The process of computing this histogram at each node is referred to as "filling" the forest, and is performed using all of the pixels in each training image. Each pixel is categorized

using the forest, and the nodes it reaches recorded in H_n.

In *partial supervision*, we do not have the pixel label c but rather each image has associated with it merely a set of categories without a mapping between those categories and the pixels. In other words, we have just a distribution $P(C|X = x)$ where x is an observed topic for the image. As we have no data about $P(p|C)$, it is modeled as a uniform distribution. Thus, to create training points to use in a partially supervised forest we first sample a category using $P(C|X = x)$ and then sample a pixel using $P(p|C)$. The forest is subsequently trained on these points, and the result has quite a low pixel accuracy, as is to be expected. However, this pixel accuracy is still greater than random chance.

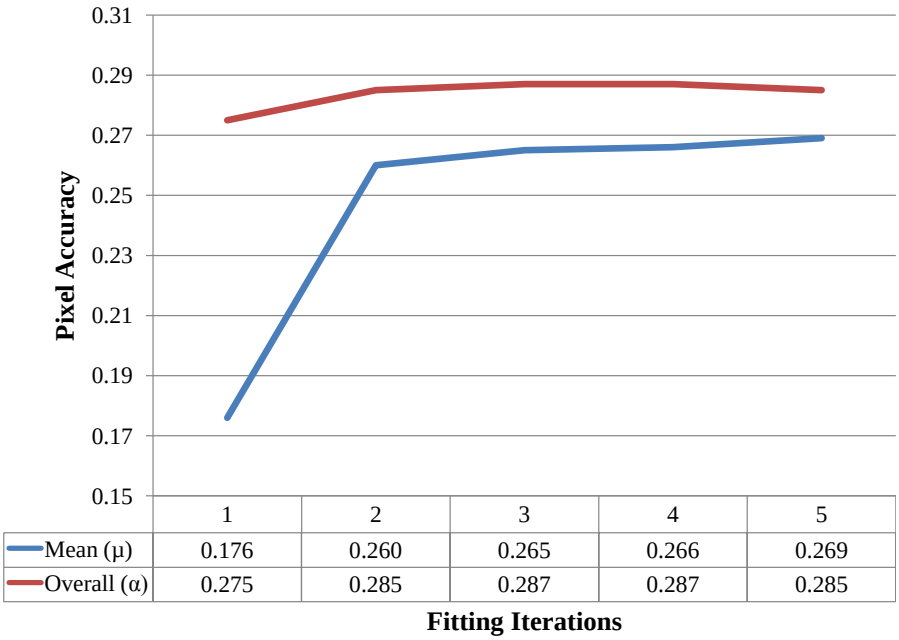
One way to think of a semantic texton forest is as a biased pixel clustering tool. Though the tree uses the pixel labels to bias the clustering process to create homogenous clusters, the pixels in a cluster will always have the same appearance. Thus, the semantic texton forest has already clustered pixels based on appearance, and the problem with which we are presented is to find the correct labels for each cluster.

Therefore, without retraining the forest, we can relabel the pixels which are clustered in each leaf node. We do this by returning to the images, as we now can calculate $P(p|C)$ as

$$P(p|C) = \frac{P(C|p)P(p)}{\sum_p P(C|p)P(p)} \,. \tag{3.3}$$

where $P(p)$ is a uniform prior over pixels in the image and $P(C|p)$ is calculated as $P(C|L(p))$.

If we then sample points based upon this new distribution and use them to fill the tree, we will have more accurate labels for the pixels in each node cluster. We can then repeat this process until the pixels in a leaf cluster have more accurate labels, using the constraints of $P(C|X)$ to inform the process. Figure 3.6 shows this process, graphing pixel categorization accuracy against rounds of training. As can be seen, it converges quite quickly, and results



	1	2	3	4	5
Mean (μ)	0.176	0.260	0.265	0.266	0.269
Overall (α)	0.275	0.285	0.287	0.287	0.285

Figure 3.6: *Partially Supervised Accuracy Improvement*. In the partially-supervised training case with semantic texton forests, the lack of pixel labels results in very low overall pixel categorization accuracy after initial tree training. However, using the tree fitting process described in the text, the forest is able to better model $P(C|n)$, resulting in a significant improvement in accuracy with only a few iterations. The numbers reported are the mean accuracy μ and the overall accuracy α, which are described in Section 3.2.2.

in a significant improvement in overall and average pixel accuracy.

3.2 Training the Forest

Training a randomized decision forest involves several parameters, namely:

Number of Trees

The number of trees in the forest is a tradeoff between accuracy and speed

Maximum Tree Depth

There are a range of different pixel tests which can be performed

Test Pool Size

Since there are too many possible tests which can be tried, the size of the subset which

is proposed at each node can play a role in how well the tree generalizes from the training data.

Percentage of Training Data per Tree

In order to create trees which are different enough to add information and capture subsets of behavior in the dataset, it is sometimes useful to hold back training data from a tree when it is being formed, though the entire dataset is always used to estimate $P(C|n)$ through the tree filling process.

Value of d

The size of the window around each pixel d can effect whether the tree learns local image characteristics or contextual information.

Tests Used

The types of tests used can play a significant role in tree training and performance, with different combinations of tests acting in a complimentary manner.

Of these, the optimal type and number of tests depends quite heavily on the nature of the dataset and the task to be performed. In the interest of discovering the best parameters for the task of pixel-level category inference (as described above) we have designed a series of experiments where the parameters are changed and their effects measured. Naturally, the cost in time and resources of performing a full exploration of the parameter space would be prohibitive, and as such we have limited our exploration to the effect one variable can have while the others are held constant.

Of particular interest are the types of test domains which we make available to the training algorithm. $A[c_0]$ and $B[c_1]$ are the values of pixels within a patch of size $d \times d$ (as seen in Figure 3.2) centered on a pixel. The channels c_0 and c_1 do not have to be the same, and indeed allowing them to be different results in a small improvement in performance. The

pixel tests below are those we used in our experiments:

Label	Test Domain
1	$A[c_0]$
2	$\log(A[c_0])$
3	$A[c_0] + B[c_1]$
4	$A[c_0] - B[c_1]$
5	$\lvert A[c_0] - B[c_1] \rvert$
6	$A[c_0] \log(B[c_1])$
7	$A[c_0] \times B[c_1]$
8	$A[c_0]/B[c_1]$

In addition to these we use two rectangle-based features: the Haar-like features of [104] and the rectangle sum features of [92], computed efficiently using integral images.

3.2.1 Building a Tree

The training data consists of a set of pixels P sampled from training images at a rate of every 4 pixels in each direction (ignoring pixels marked as background). We subsample pixels in order to decrease the time required for tree construction, but use all pixels to fill the tree after construction as described in Section 3.1.1, which corrects for any overfitting caused by building the tree on a subset of the data. We construct a tree by recursively splitting P into two subsets $P_{left} \cup P_{right} = P$ based upon a randomly sampled test. P_{left} is sent to the left child and P_{right} is sent to the right child and the process is repeated until a stopping condition is met. We choose the test used to split P in the same manner as [62], that is to say by choosing which test in a randomly sampled pool of tests results in the largest expected

gain in information about the node categories, calculated as

$$\Delta E = -\frac{|P_{left}|}{|P|}E(P_{left}) - \frac{|P_{right}|}{|P|}E(P_{right}) \,, \tag{3.4}$$

where $E(P)$ is the Shannon entropy of the classes in the set of examples P. In our experiments, the set of tests is bounded to be one or more of the test domains listed above. However, there is no reason why the test could not be something entirely different, *e.g.* a function of a feature descriptor computed at the pixel.

The weakness of this depth-first method of tree construction is that only one attempt is made to split a node. Due to the random nature of the feature sampling process this means that in some cases the best test, chosen using the measure above, is suboptimal. This can be avoided by requiring that a split only be performed if the entropy gain is above a certain threshold, but then an entire branch can be prematurely pruned, resulting in a lopsided and ineffectual classifier. The way to avoid this is to build the tree in a breadth-first manner. Instead of recursively splitting nodes until a maximum depth is reached, each current leaf-node is tested with the current sample of tests and the entropy gain measured. Those above a threshold are split, with the process repeating until no leaf nodes can be split (either due to no entropy gains over the threshold or all nodes being at maximum depth). The advantage of this method is that problematic leaf nodes are given several chances to be split, increasing the likelihood that an optimal test will be chosen, at the cost of introducing an arbitrary threshold on the entropy gain.

Learning Invariances

Using the pixels of each image in the training set gives a good estimate of the true distribution $P(C|n)$ for each node once a tree is built. Once a tree is constructed, a "filling process"

is performed by which every pixel in every training image is categorized by the forest and a record kept of which nodes it reaches. After this is complete, the counts are normalized (with a Dirichlet prior over categories taken into account), thus ensuring that a tree has seen all of the data in the training dataset even if it is trained on a subset of the data.

Unseen images, though, are very likely to be different in many ways from the training images, thus causing inaccuracy in the pixel classification. In other methods, this effect is lessened through the use of an invariant feature descriptor, as discussed in Appendix A. Semantic texton forests, however, use the raw pixels, and so must be made invariant by other means. Thus, as we estimate the distribution $P(C|n)$ with the training images we also augment these images with copies that are artificially transformed geometrically and photometrically as done by Lepetit in [62]. Laptev observed in [60] that augmenting training data in this way results in a significant improvement in recognition performance, and indeed our results mirror his in this regard. This allows the forest to *learn* the right degree of invariance required for a particular problem. In our experiments we used rotation, scaling, and left-right flipping as geometric transformations, and affine photometric transformations.

3.2.2 Experiments

In our experiments, accuracy is measured as the mean percentage of pixels labeled correctly over all categories. As in Chapter 2, we compute a confusion matrix M, but instead of it being over images $I \in R$ it is over pixels $p \in P_R$, where P_R is the set of test pixels. Thus, the individual cells of M are computed as

$$M[i,j] = |\left\{p : p \in P_R, G(p) = c_i, \text{argmax}_c P(c|L_p) = c_j\right\}| \tag{3.5}$$

For these experiments we report the mean category accuracy μ, calculated as

$$\mu = \frac{1}{|\{c_i\}|} \sum_{c_i} \alpha_{c_i} \tag{3.6}$$

where

$$\alpha_{c_i} = \frac{M[i,i]}{\sum_j M[i,j]} \,, \tag{3.7}$$

and the overall accuracy α, calculated as

$$\alpha = \frac{\sum_i M[i,i]}{\sum_i \sum_j M[i,j]} \,. \tag{3.8}$$

μ gives a sense of overall categorization, as the good performance of categories which are represented by many pixels (*e.g.* sky, water, grass) does not mask out the poor performance of those which inhabit fewer, as is the case with the overall accuracy α. α, on the other hand, gives a sense of what proportion of the image the system can reliably segment. Both are important to get a sense of overall performance of the system, *e.g.* a high α and low μ indicates overfitting to a particular category which is disproportionately represented in the dataset.

We selected as our control training scenario the following parameters:

Parameter	Value
Number of Trees	5
Maximum Depth	10
Feature Tests	A, $A + B$, $A \log(B)$, $\lvert A - B \rvert$
Pool Size	400
d	10
Channels	CIELab
Data %	25

In each experiment, we change a single training parameter while keeping all others constant to see the effect it has on test performance. Experiments were performed on the

MSRC21 dataset (for further details, see Appendix B). In each experiment, ten trees were trained on a subset of the data and then filled with all of the data points as described in Section 3.2.1. Ten forests of five trees (with the exception of the number of trees experiment) were then created from permutations of these ten trees. The reported values are the mean α and μ of 10 segmentation trials run with those forests over the test data after being trained on the training and validation data with the specified parameters. Error bars indicating the standard error are omitted due to the remarkably low error on the values making them indiscernible. It is quite possibly due to the fact that the different forests being used for each trial are 5 trees chosen from the same set of 10, but even so we were very intrigued to note that 10 different trees trained independently on different subsets of the data but with the same parameters achieved more or less the same accuracy on the test data.

In Figure 3.7, we show the effect different combinations of feature tests has on segmentation performance. In each of the five trials we chose a different random order of five feature tests, shown in Table 3.1, and in each step added an additional feature for the tree to use and increased the feature pool size accordingly. One trial was done with every test and a pool size of 1000 (the rightmost bar on the graph) showing the practical limit on accuracy. Every additional test made available to the algorithm will usually improve performance, but there are certain groups of tests which work together better than others. So it is that performance can actually drop, sometimes dramatically, when a new test is added that elsewhere when added improved performance. The ideal mixture of tests depends to a certain extent on the data, and these experiments should only be considered in so far as they show the rate at which adding different feature tests improves performance. One good method of finding out which tests work best for a dataset is to do as we did and to train one forest with a large pool size and every available test. As each tree will choose tests based purely on information gain, this will give a very good indicator of which tests work best for the dataset, and

Trial	1	2	3	4	5
1	A	$A \times B$	$A - B$	Haar	$\|A - B\|$
2	$\|A - B\|$	$A - B$	A/B	A	$\log(A)$
3	A/B	$A - B$	$\log(A)$	$A \times B$	A
4	$\log(A)$	Rectangle	$A - B$	$\|A - B\|$	$A + B$
5	$A \times B$	A	$A + B$	$A \log(B)$	A/B
Pool Size	100	200	300	400	500

Table 3.1: *Test Domain Experimental Setup*. The size of the feature pool and the number of different feature domains those features could be drawn from increases from right to left, with 25 total trials being performed. Results are shown in Figure 3.7.

	Counts	%
Rectangle	670930	39.89%
Haar	316368	18.81%
A	179816	10.69%
$A + B$	118638	7.05%
$A \times B$	107146	6.37%
$\|A - B\|$	105794	6.29%
$A \log(B)$	92274	5.49%
$A - B$	45968	2.73%
A/B	44954	2.67%
Total	1681888	

Table 3.2: *Test Proportions for MSRC21 Dataset*. We trained a semantic texton forest on the MSRC21 dataset, making all of the tests in the table available to it during training. We then recorded the counts for each test over the entire forest to get a sense of which tests were most useful for categorization. As can be seen, the rectangle-based features performed best and were chosen at a much higher rate than the others. It is important to note, however, that the Haar-like features are a superset of the additive and subtractive pixel features.

a subset can be chosen either for quicker training or to optimize tree performance depending on the situation at hand. As an example, Table 3.2 gives the proportions of tests for the forest which had every test available to it, showing that for the evaluation dataset rectangle features like the rectangle sum and Haar-like features perform very well.

In Figure 3.8 we show the effect of the number of trees in the forest. As can be seen, there are diminishing returns as forest size increases. In order to investigate the effects of different methods of representing color on performance we trained forests on grayscale, RGB, CIELab, and HSV images, the results of which can be seen in Figure 3.9. As can be seen, CIELab results in the best performance, most likely due to its useful features (*e.g.* meaningful perceptual distances, device independence) for computer vision as described in

[50]. We show the effect of different tree depths in Figure 3.10. The values chosen for this trial were dependent on the dataset size, as the amount of data needed to fill a tree increases exponentially with tree depth, but it can be seen that as the number of nodes in the tree increases so does its classification ability, which is to be expected due to the way in which the trees are trained (*i.e.* nodes will stop splitting if there is no expected information gain). Finally, the effect of the d parameter (the size of the window test pixels can be chosen from) can be seen in Figure 3.11. The effect here of a steady increase in average accuracy coupled with an increase and then decline in overall accuracy is very interesting. We suspect it is because the easier categories which take up many of the pixels in the dataset (*e.g.* grass and sky) do not require the context of nearby pixels to be classified, but the smaller and more complex categories can take advantage of nearby information to aid in classification that is only possible when the window of possible pixels is significantly increased.

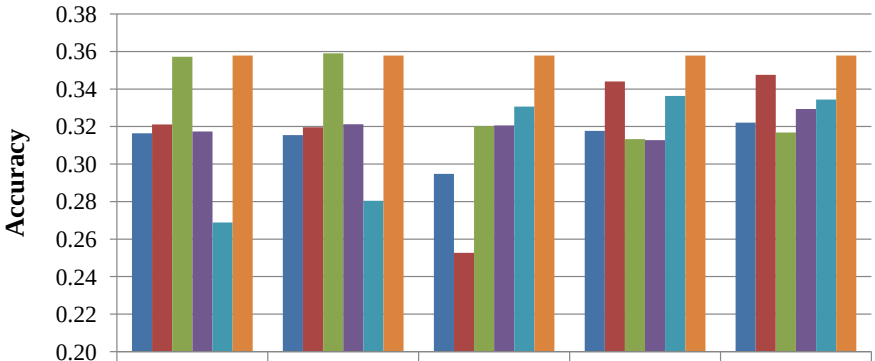
3.3 Image Categorization

Let us revisit the bag-of-words model for image categorization from Chapter 2. As an alternative to the typical method for creating these histograms, we propose the localized bag of semantic textons (BOST), illustrated in Figure 3.12. This extends the bag of words with low-level semantic information, as follows.

Given for each pixel p the leaf nodes $L(p) = (l_1, \ldots, l_T)$ and inferred class distribution $P(C|L(p))$, one can compute over image region r

1. A non-normalized histogram $H_r(n)$ that concatenates the occurrences of tree nodes n across the different trees [72], and

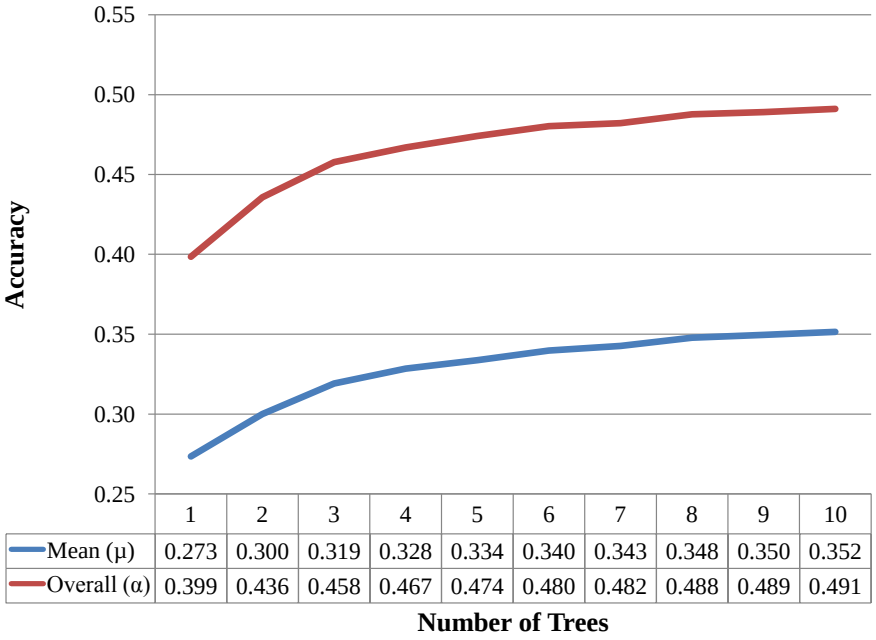
2. A conditional distribution over the region given by the average class distribution $P(C|r) = \sum_{p \in r} P(C|L_p)P(p)$.



	1	2	3	4	5
Tests 1	0.316	0.315	0.295	0.318	0.322
Tests 2	0.321	0.320	0.253	0.344	0.348
Tests 3	0.357	0.359	0.320	0.313	0.317
Tests 4	0.317	0.321	0.321	0.313	0.329
Tests 5	0.269	0.280	0.331	0.336	0.334
Tests All	0.358	0.358	0.358	0.358	0.358

Sets of Tests

Figure 3.7: *Effect of Different Test Domains*. The values shown are the mean over ten trials of the mean category accuracy (μ). The computation for μ is described in Section 3.2.2. This graph tracks the effect of increasing the test domains on accuracy, with the overall trend being that the larger the domain of tests the more accurate the system becomes, though there is quite a lot of variation with some compositional changes, particularly in set 3. For reference, a system trained with all possible tests and a pool size of 1000 is shown as the rightmost bar. The meanings of the numbers in the graph are explained in Table 3.1.



	1	2	3	4	5	6	7	8	9	10
Mean (μ)	0.273	0.300	0.319	0.328	0.334	0.340	0.343	0.348	0.350	0.352
Overall (α)	0.399	0.436	0.458	0.467	0.474	0.480	0.482	0.488	0.489	0.491

Figure 3.8: *Effect of Increasing the Number of Trees*. The values shown are the mean over ten trials of the overall per-pixel accuracy (α) and the mean category accuracy (μ). The computations for α and μ are described in Section 3.2.2. We see here a clear logarithmic growth in performance with forest size, with the elbow of the graph occurring at 5 for this dataset.

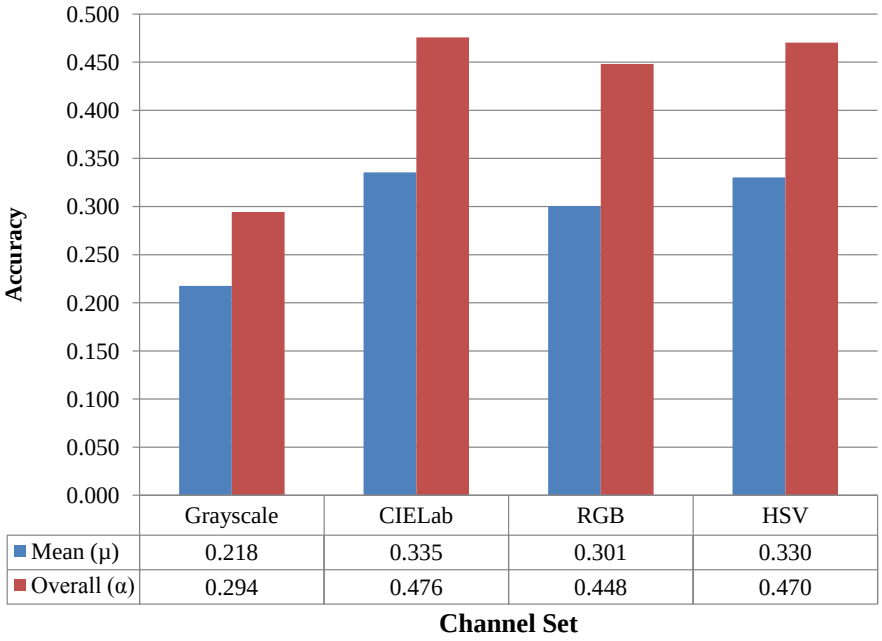
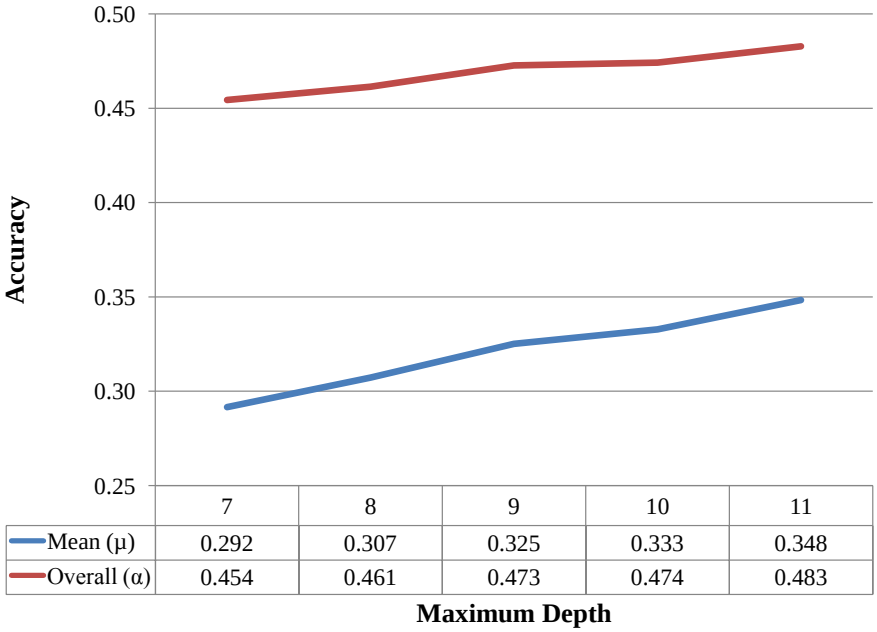
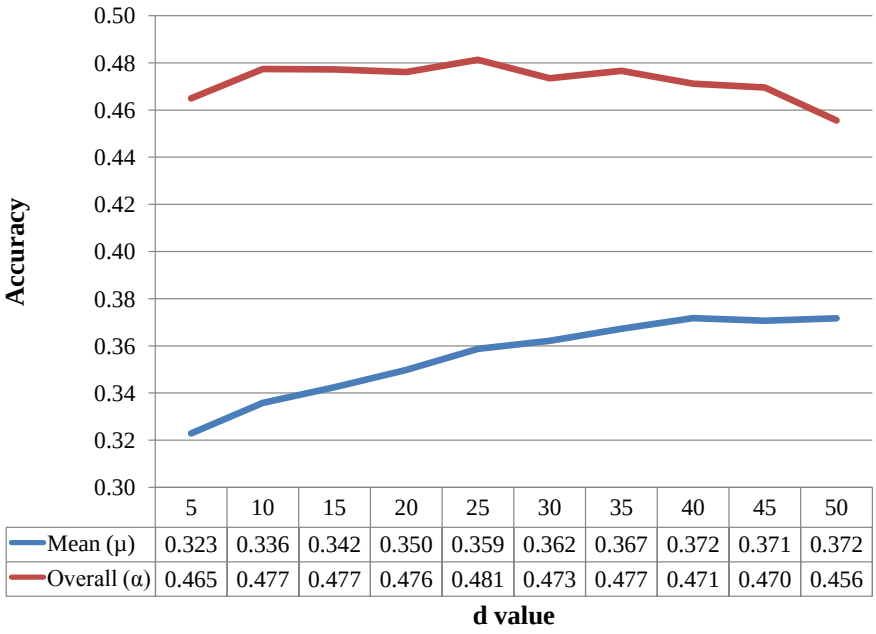


Figure 3.9: *Effect of Different Channels*. The values shown are the mean over ten trials of the overall per-pixel accuracy (α) and the mean category accuracy (μ). The computations for α and μ are described in Section 3.2.2. There is a slight but consistent advantage to be gained by using orthogonal color spaces over RGB, and the addition of color gives a significant improvement over grayscale, as is to be expected.



	7	8	9	10	11
Mean (μ)	0.292	0.307	0.325	0.333	0.348
Overall (α)	0.454	0.461	0.473	0.474	0.483

Figure 3.10: *Effect of Maximum Depth*. The values shown are the mean over ten trials of the overall per-pixel accuracy (α) and the mean category accuracy (μ). The computations for α and μ are described in Section 3.2.2. We see here that tree depth results in a linear growth in accuracy, particularly for class average accuracy. While the discriminative power of the forest increases with maximum depth, so does the possibility of overfitting to the data. This can be avoided by only training on subsets of the data, but one is faced with the problem of having enough data to fill the tree so as to model the correct uncertainty (as the amount of data required increases exponentially with each additional level)



	5	10	15	20	25	30	35	40	45	50
Mean (μ)	0.323	0.336	0.342	0.350	0.359	0.362	0.367	0.372	0.371	0.372
Overall (α)	0.465	0.477	0.477	0.476	0.481	0.473	0.477	0.471	0.470	0.456

Figure 3.11: *Effect of d.* The values shown are the mean over ten trials of the overall per-pixel accuracy (α) and the mean category accuracy (μ). The computations for α and μ are described in Section 3.2.2. We see here a general trend by which as the parameter d increases class average accuracy increases, but overall accuracy declines. Further experiments beyond the value of 55 were inconclusive due to the effect of less data being available to the training process, but seem to support this trend.

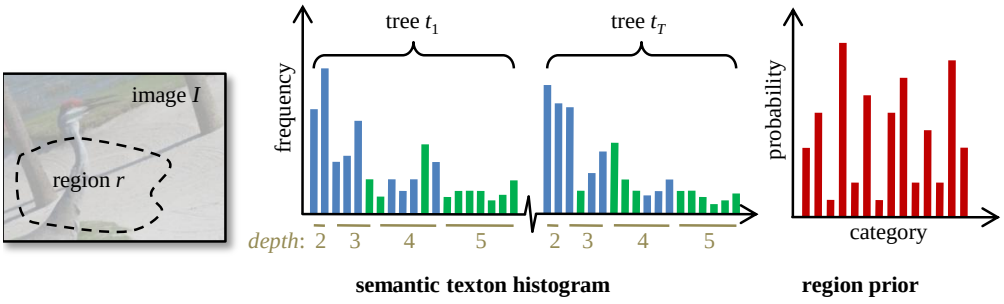


Figure 3.12: **Bags of semantic textons.** Within a region r of image I we generate the semantic texton histogram and region prior. The histogram incorporates the implicit hierarchy of clusters in the STF, containing both STF leaf nodes (green) and split nodes (blue). The depth d of the nodes in the STF is shown. The STFs need not be to full depth, and empty bins in the histogram are not shown as the histogram is stored sparsely. The region prior is computed as the average of the individual leaf node category distributions $P(C|l)$.

We perform experiments with tree histograms (unlike the leaf histograms of [72]) where we include both leaf nodes l and split nodes n in the histogram, such that

$$H_r(n) = \sum_{n' \in \text{child}(n)} H_r(n') \ . \tag{3.9}$$

This histogram therefore uses the hierarchy of clusters implicit in each tree. Each $P(C|L(p))$ is already averaged across trees, and hence there is a single region prior $P(C|r)$ for the whole forest.

3.3.1 Tree Histograms and Pyramid Matching

Consider first the BoST histogram computed for just one tree in the STF. The kernel function (based on [37]) is then

$$K(P,Q) = \frac{1}{\sqrt{Z}} \tilde{K}(P,Q) \ , \tag{3.10}$$

where Z is a normalization term for images of different sizes computed as

$$Z = \tilde{K}(P,P)\tilde{K}(Q,Q) \ , \tag{3.11}$$

and $\tilde{K}$ is the actual matching function, computed over levels of the tree as

$$\tilde{K}(P,Q) = \sum_{d=1}^{D} \frac{1}{2^{D-d+1}} (\mathcal{I}_d - \mathcal{I}_{d+1}) \ , \tag{3.12}$$

using the histogram intersection $\mathcal{I}$ [97]

$$\mathcal{I}_d = \sum_j \min(P_d[j], Q_d[j]) \ , \tag{3.13}$$

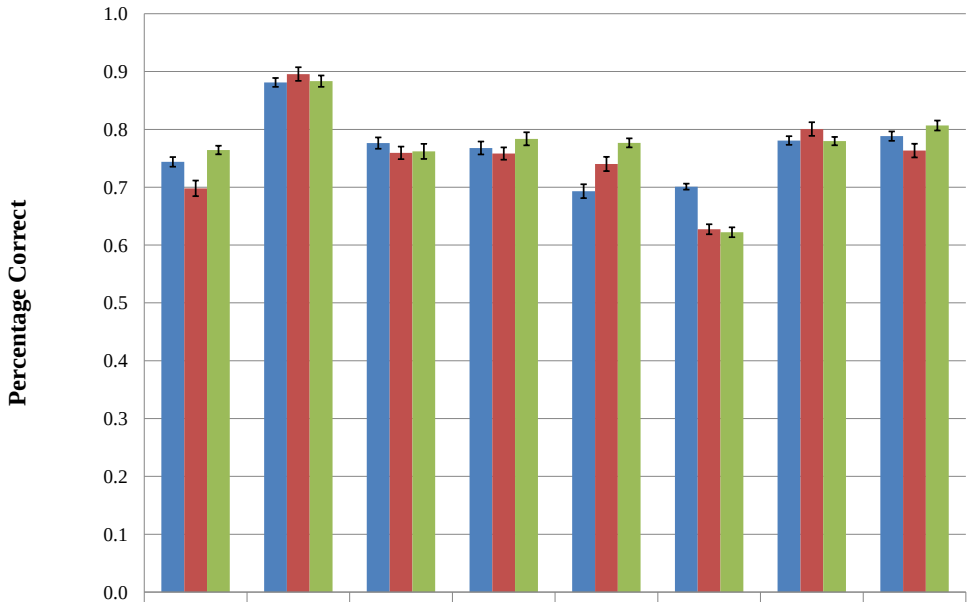
where D is the depth of the tree, P and Q are BoSTs, and P_d and Q_d are the portions of the histograms at depth d, with j indexing over all nodes at depth d. There are no nodes at depth $D+1$, hence $\mathcal{I}_{D+1} = 0$. If the tree is not full depth, missing nodes j are simply assigned $P_d[j] = Q_d[j] = 0$.

3.3.2 Results

In these experiments, we trained a forest using the following parameters, selected based on the results of the experiments in Section 3.2.2:

Parameter	Value
Number of Trees	5
Maximum Depth	10
Feature Tests	A, $A+B$, $A\log(B)$, $\lvert A-B\rvert$, Rectangle, $A-B$
Pool Size	600
d	10
Channels	CIELab
Data %	25

The dataset used was the Scenes database (see Appendix B), chosen for the purpose of comparison with the technique of the last chapter. These experiments are set up in the same way, but instead use BoSTs and and pyramid match kernel for training the SVM. As can be seen, the semantic texton forests achieve the same or better performance than the techniques from the previous chapter. What is worth mentioning is that it is able to combine all of the same cues as those techniques, but can compute the BoST for an image very efficiently. Due to the complexity of the process ($O(n \log n)$) it is feasible for this system to perform categorization at frame rate, which would be difficult to achieve for the techniques in Chapter 2.



	coast	forest	highway	inside city	mountain	open country	street	tall building
STF	0.744	0.881	0.776	0.768	0.693	0.701	0.781	0.788
SVM	0.698	0.896	0.759	0.758	0.740	0.627	0.801	0.763
Joint Boosting	0.764	0.883	0.762	0.784	0.777	0.622	0.780	0.807

Figure 3.13: *Scene Categorization Results*. The values shown are image accuracy per category as in Section 2.4. We trained an SVM using the pyramid match kernel and BOSTs (STF-SVM), and compare the performance against the Cross-Cue Joint Boosting and SVM classifiers from Chapter 2 on the scene dataset. We use the same train and test splits, and average over them. As can be seen, the technique is on par, if not slightly better in some cases, than these other bag-of-words based algorithms.

3.4 Summary

Semantic texton forests are a promising new image description framework, being very efficient to compute and able to function on regions of arbitrary size and shape within the image. Additionally, the hierarchical structure of the trees in the forest provides a rich tool for comparison between descriptors. Perhaps more intriguing, however, is the ability of the STF to perform a *semantic segmentation*, that is to infer a category label for every pixel in the image. In the next chapter we will explore how to refine the segmentation provided by a STF and achieve state-of-the-art performance for the semantic segmentation of images.

CHAPTER 4

SEMANTIC SEGMENTATION

The idea of the semantic segmentation of an image is built on a model of image generation which is based on dividing the real-valued and continuous visual signal of an image into *cells*, or a regular rectangular sample grid, each of which is sufficiently explained by an underlying label. To perform a semantic segmentation of an image is to infer the semantic label for every cell. For example, look at the image in Figure 4.1. Using simple semantic labels, the pixels in the image have been explained, each one generated by some unknown model for the category label. If such a segmentation can be achieved, then the image can be catalogued for image search, used for navigation, or any number of other tasks which require basic semantic understanding of arbitrary scenes.

4.1 Soft Classification of Pixels

Our formulation of semantic segmentation centers on the cell model, presented graphically in Figure 4.2 (graphical models are a common method of visualizing joint distributions over several random variables, see Bishop [8]). In this model, for every image there is a random variable X whose value represents the subject matter, or broad image-level category, of the image (*e.g.* the forest, an office, the moon). The likelihood of the various image-level categories are governed by some learned parameter χ. This topic generates cell labels C_i for

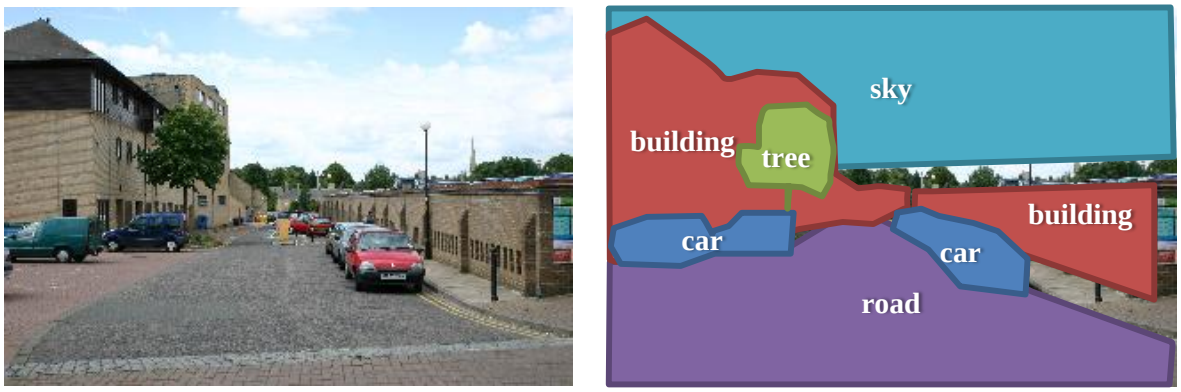


Figure 4.1: *Semantic Segmentation*. A semantic segmentation of an image is one which groups the pixels together by common semantic meaning. Shown is one such segmentation of an image, using as pixel labels the objects in the scene.

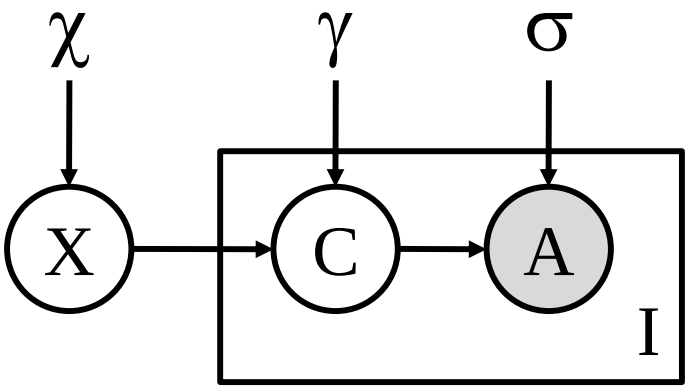


Figure 4.2: *Cell-based Image Generation Model*. This figure is a graphical representation of the model [8]. For each image we choose a particular topic, represented by the random variable X. These can be thought of as scenes (*e.g.* the forest, an office, the moon). For a particular topic, we generate I cells on a grid, where each cell has a semantic category C that generates the appearance A. To perform a semantic segmentation, we infer the semantic category for each grid cell.

each grid cell i. These are general semantic categories, that is categories of object or entity (*e.g.* trees, computers, rocks). The conditional probability of a semantic category given the image-level category is governed by the learned parameter γ. Finally, we have the appearance of a cell (*e.g.* a pixel patch, a descriptor, a histogram) which is generated by the cell category using some process which adds in some noise (indicated by σ), *e.g.* a normal distribution over intensity values.

Naturally, a normal distribution over intensity values is not a sufficient generative patch model. While a true generative patch model may be out of reach in the near future, the promising jigsaw model [61] and other emerging techniques utilizing deep inference are showing great promise of achieving this enviable goal. In the meantime, however, the best results have been obtained via discriminative methods. Since the appearance is observed, we can infer the category label from the appearance (essentially, reverse the arrow between C and A) and infer C by marginalizing over the topics and incorporating a discriminative model for $P(C|A)$. In Chapter 3 we discussed semantic texton forests and their ability to estimate a distribution over a set of labels for arbitrary pixel regions in a discriminative manner. In this chapter we will discuss how they can be used to infer $P(C|A)$ at the level of an image grid cell.

We have already established that we can infer $P(C|A)$ for cells at the level of a pixel with a semantic texton forest:

$$P(C|A_p) = P(C|L(p)) \propto \sum_{t=1}^{T} P(C|l_t)P(t) \; . \tag{4.1}$$

As the size of a cell increases, we are faced with the problem of agglomerating the informa-

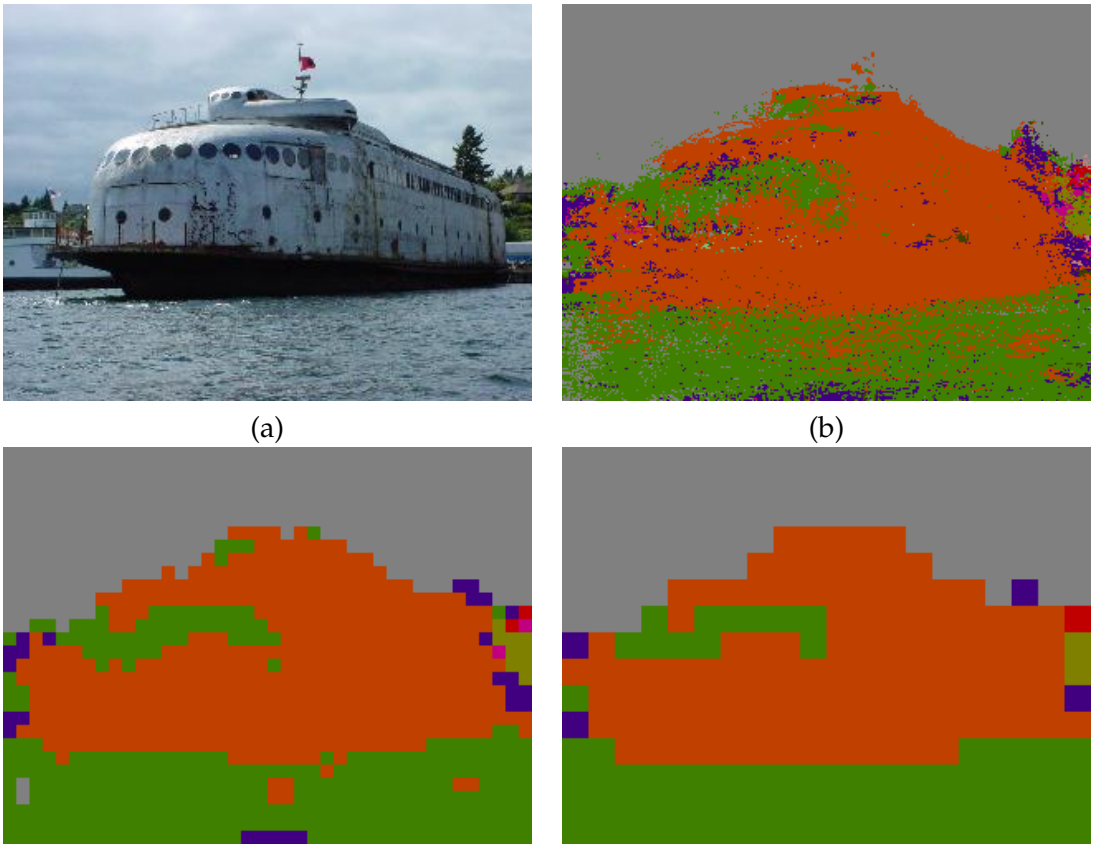


Figure 4.3: *Cell Segmentation*. (a) is the original image. (b) is the image with pixel-level maximum *a posteriori* (MAP) labeling. In (c), we subsample the distributions from (b) by 8, and in (d) the image in (b) has been subsampled by 16. As grid square size increases, detail is lost but the overall accuracy of the segmentation increases.

tion held in individual pixels to regions. We can calculate this as

$$P(C|A_r) = P(C|r) \propto \sum_p P(C|L(p))P(p|r) \;, \tag{4.2}$$

which leaves the conditional distribution $P(p|r)$ to be modeled. There are two practical choices for this. The first is a bivariate normal distribution with diagonal covariance and centered on the cell, and the second is

$$P(p|r) = \begin{cases} \frac{1}{|P_r|} & p \in P_r \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

where P_r is the set of pixels in a region. Figure 4.3 depicts both the pixel- and cell-level maximum *a posteriori* labelings for an image.

So far, we have discussed the conditional distribution $P(C|A)$, but we have yet to touch on $P(C|X)$ or $P(X)$. In the following section, we will discuss how it is possible to again use semantic texton forests to infer the parameter χ from the image data.

4.2 Image-level Semantic Constraints

Using a inferred image topic or scene like X to aid in the recognition of objects in the image can result in significant improvement, as found by Russell *et al.* in [83]. We use it here for precisely that purpose: to use global image understanding to disambiguate local image information. The inference of $P(X)$ is essentially the task of image categorization, as discussed in Chapter 2. The task of categorizing an image consists of determining those categories (*e.g.* forest images, office images, moon images) to which an image belongs. There has been much research performed on this problem, with the most successful of previous approaches using global image information [77], bags of words [29] or textons [106]. We extend the STF cat-

egorization algorithm presented in Chapter 3 to exploit not just the hierarchy of semantic textons but also the node prior distributions $P(C|n)$. We still use a non-linear support vector machine (SVM), which depends on a kernel function K that defines the similarity measure between images. To take advantage of the hierarchy in the STF, we adapt the innovative pyramid match kernel [37] to act on a pair of BoST histograms computed across the whole image, computed in the same way as described in Chapter 3.

The kernel over all trees in the STF is calculated as $K = \sum_t \kappa_t K_t$ with mixture weights κ_t. Similarly to [110], we found $\kappa_t = \frac{1}{T}$ to result in the best categorization results. This method is very effective, but can be improved by using the learned distributions $P(C|n)$ in the STF. If $P(c|n)$ is large for class c and node n, then a large count of node n is a strong indicator that class c is present in the image. If $P(c|n)$ is small, you gain less information since the likely presence of other categories only helps as context. For example, if you are looking for grass in an image, you care more about the count of nodes likely to be grass than those likely to be motorbike. Following this intuition, we build a 1-vs-others SVM kernel K_c per category, in which the count for node n in the BoST histogram is weighted by the value $P(c|n)$. This helps balance the categories, by selectively down-weighting those that cover large image areas (*e.g.* grass, water) and thus have inappropriately strong influence on the pyramid match, masking the signal of smaller classes (*e.g.* cat, bird).

In our experiments, we show the improvement that the pyramid match kernel on the hierarchy of semantic textons gives over a radial basis function on histograms of just leaf nodes. We also obtain an improvement using the per-category kernels K_c instead of a global kernel K.

	Global kernel K	Per-category kernel K_c
RBF	.499	.525
MK	.763	**.783**

Table 4.1: **Image categorization results.** (Mean AP).

4.2.1 Categorization Results

The mean average precisions (AP) in 4.1 compare our modified pyramid match kernel (PMK) to a radial basis function (RBF) kernel, and compare the global kernel K to the per-category kernels K_c. In the baseline results with the RBF kernel, only the leaf nodes of the STF are used, separately per tree, using term frequency/inverse document frequency to normalize the histogram. The PMK results use the entire BOST which for the per-category kernels K_c are weighted by the prior node distributions $P(c|n)$. Note that the mean AP is a much harder metric and gives lower numbers than recall precision or AuC; the best result in the table shows very accurate categorization. For a description of how this quantity is computed, see Chapter 7. As can be seen, the pyramid match kernel considerably improves on the RBF kernel. By training a per-category kernel, a small but noticeable improvement is obtained. Due to its performance, we use the PMK with per-category kernels to train the SVM used as χ. The full categorization results are presented as precision recall curves in Figure 4.4.

4.2.2 The Image Level Prior

To relate this system of image categorization back to the cell model for semantic segmentation, the task the SVM is performing is to infer the value of X for an image. In essence, χ is the support vector machine as opposed to being a global prior distribution over topics, and depends on the BOST computed for a particular image in order to compute $P(X)$. We can

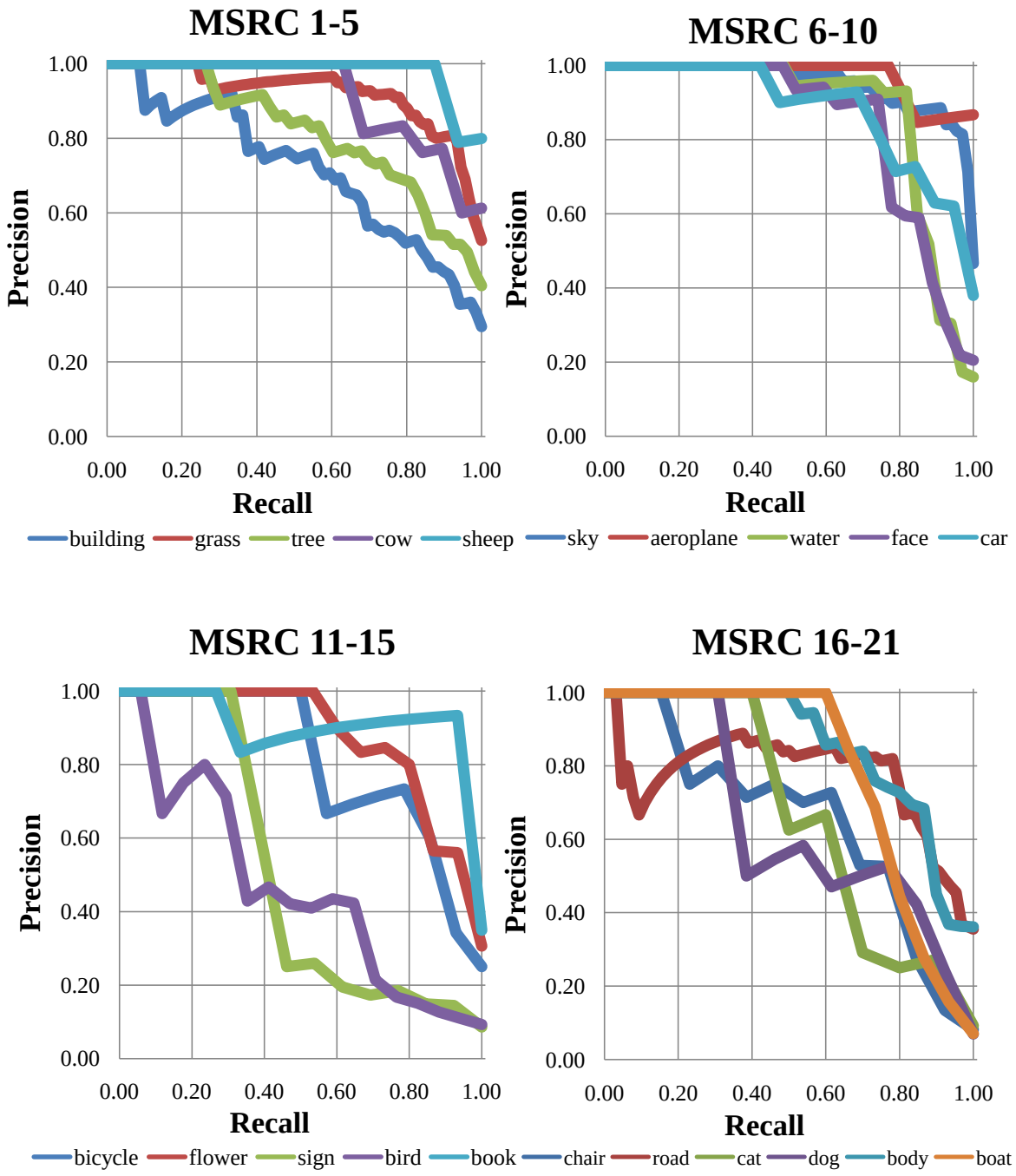


Figure 4.4: *MSRC Categorization Results* Shown here are the precision recall curves for all of the categories in the MSRC21 dataset.

think of $P(X)$ as being related to χ thus:

$$P(X = x) = \chi(\text{BoST}_d)[x] \tag{4.4}$$

where χ is the SVM classifier and x is a value of the random topic variable X.

Moreover, since X and C in the scenario we have just presented have the same domain, γ is not a learned distribution but instead a binary function:

$$\gamma[x, c] = \begin{cases} 1 & \text{if x = c} \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

Thus, whereas we would usually marginalize over X when computing $P(C_i)$ in the following manner (substituting lower case letters for observed variables):

$$P(C_i = c_i) = P(C_i = c_i|A_i = a_i) \sum_X P(X)P(C_i = c_i|X) \prod_{j \neq i} \sum_{C_j} P(C_j|X)P(C_j|a_j) \tag{4.6}$$

given the binary nature of $P(C|X) = \gamma[x, c]$, this reduces to:

$$P(C_i = c_i) = P(C_i = c_i|A_i = a_i)P(X = c_i) \prod_{j \neq i} P(C_j = c_i|A_j = a_j) \tag{4.7}$$

where $P(X = c_i)$ is the result of the SVM classifier and $P(C = c|A = a)$ is computed using the STF as described above. Furthermore, we approximate the effect of the right-hand product by a power α on $P(X = c)$ in the results presented in this chapter. This was done to provide a variable by which the system could be optimized, and also to aid with computational efficiency. Since this is no longer inference *per se*, we refer to the result of this process as an "image level prior", or ILP. While these optimizations result in a system which is very efficient, they are made at the cost of a more powerful and expressive model.

We explore a fully generative implementation of the cell graphical model in Chapter 5.

4.3 Compositional Constraints

To demonstrate the power of the BOST as a feature for segmentation, we adapt the TextonBoost algorithm [92]. The goal is to segment an image into coherent regions and simultaneously infer the class label of each region. In [92], a boosting algorithm selected features based on localized counts of textons to model patterns of texture, layout and context. The context modeled in [92] was 'textural', for example: sheep often stand on something green. We adapt the rectangle count features of [92] to act on both the semantic texton histograms and the BOST region priors. The addition of region priors allows us to model context based on *semantics* [80], not just texture. Continuing the example, our new model can capture the notion that sheep often stand on *grass*, as is seen in Figure 4.5.

In place of boosting, we decided to use a second randomized decision forest due to the speed of training and classification. We train this *segmentation forest* to act at image cells i, using bags and priors computed using Equation 4.3 for $P(p|r = i)$. At test time, the segmentation forest is applied at each pixel p densely or, for more speed, on a grid. The most likely class in the averaged category distribution gives the final segmentation for each cell. The split node functions compute either the count $H_{r+i}(n = n')$ of semantic texton n', or the probability $P(C = c \mid r + i)$ of class c, within rectangle r translated relative to cell i. By translating rectangle r relative to the cell i being classified, and by allowing r to be a large distance away from i (up to half the image size), such features can exploit texture, layout and context information. Our extension to their features exploits semantic context by using the region prior probabilities $P(C|r + i)$ inferred by the semantic textons.

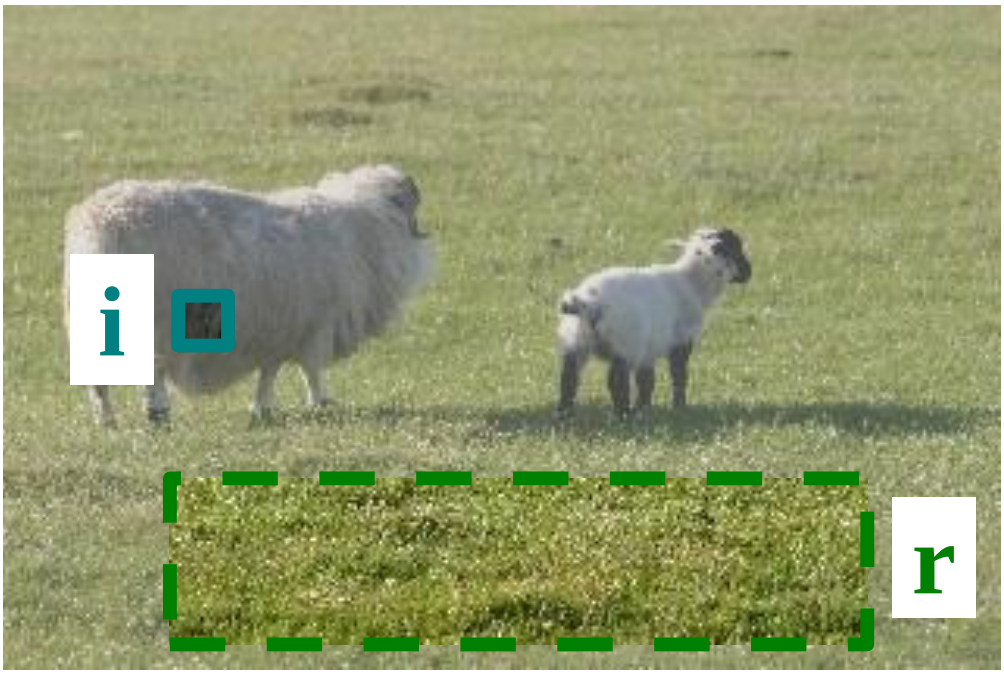


Figure 4.5: *Compositional Constraints*. For the second-level randomized decision forest, we use rectangle sum tests of the kind used in Textonboost [92]. For each image cell i the BoST and region priors of a rectangle r of a random size and offset are used as a feature vector, and the tests for the forest are thresholds on the dimensions of that vector. Thus, the system can either use leaf-node counts in a rectangle, or region priors. In this case, the region prior for grass will be quite high, and the system can learn that sheep stand on grass.

4.4 Experiments

Before presenting in-depth results for segmentation, let us look briefly at the STFs themselves. In Figure 4.6, we visualize the inferred leaf nodes $L = (l_1, \ldots, l_T)$ for each pixel i and the most likely category $c_i = \arg\max_{c_i} P(c_i|L)$. Observe that the textons in each tree capture different aspects of the underlying texture and that even at such a low level the distribution $P(c|L)$ contains significant semantic information. 4.2 gives a naïve segmentation baseline on the MSRC dataset by comparing c_i to the ground truth.

	Global	Average
supervised	48.7%	41.5%
weakly supervised	17.7%	27.8%

Table 4.2: *Naïve Segmentation Baseline on MSRC21.* Using the parameters chosen as a result of the experiments in Chapter 3 we are able to obtain a solid baseline for segmentation.

Clearly, this segmentation is poor, especially when trained in a weakly supervised manner, since only very local appearance and no context is used. Even so, the signal is remarkably strong for such simple features (random chance is under 5%). We show below how using semantic textons as features in higher level classifiers greatly improves these numbers, even with weakly supervised or unsupervised STFs.

Except where otherwise stated, we used STFs with the following parameters, hand-optimized on the MSRC validation set: distance $d = 31$, $T = 5$ trees, maximum depth $D = 10$, 500 feature tests and 5 threshold tests per split, and $\frac{1}{4}$ of the data per tree, resulting in approximately 500 leaves per tree. Training the STF on the MSRC dataset took only 15 minutes. The tests used were $A + B$, $A - B$, $|A - B|$, $A \log(B)$, A and a rectangle test, a combination motivated by the experiments in Chapter 3.

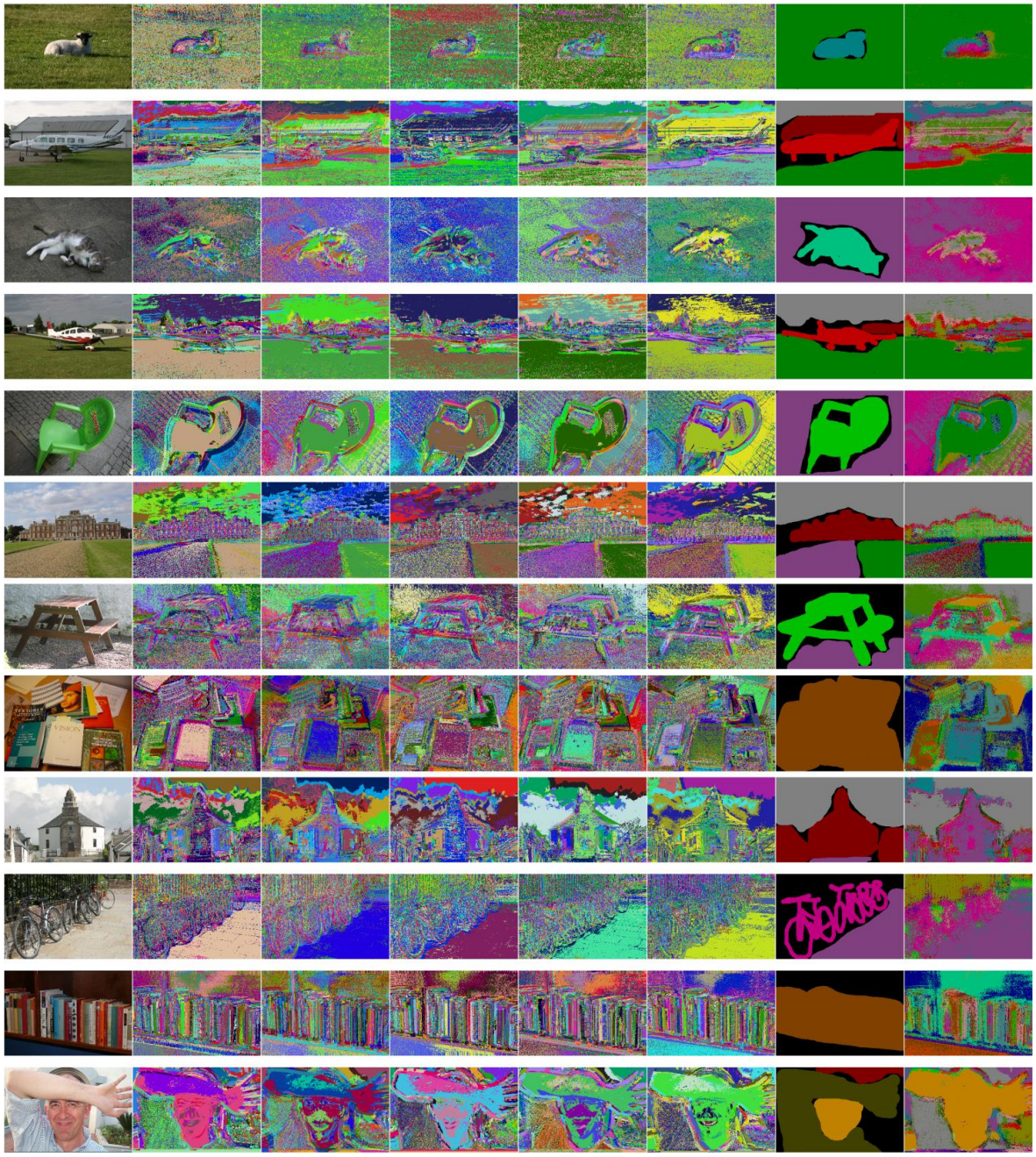


Figure 4.6: *Textonizations*. Shown here are a large selection of textonizations performed by a semantic texton forest. The first column shows the image, the middle five are textonizations from the five trees in the forest, followed by the ground truth image and the combined textonization of the forest. In the textonizations, a separate color is given to each texton index to give a sense of leaf-membership for a pixel.

4.4.1 MSRC21 Dataset

We first examine the influence of different aspects of our system on segmentation accuracy. We trained segmentation forests using (a) the histogram $H_r(l)$ of just leaf nodes l, (b) the histogram $H_r(n)$ of *all* tree nodes n, (c) just the region priors $P(c|r)$, (d) the full model using all nodes and region priors, (e) the full model trained without random transformations, (f) all nodes using an unsupervised STF (no region priors are available), and (g) all nodes using a weakly-supervised STF (only image labels). The category average accuracies are given in Table 4.3 with and without the image-level prior.

		Without ILP	With ILP
(a)	only leaves	61.3%	64.1%
(b)	all nodes	**63.5%**	65.5%
(c)	only region priors	62.1%	66.1%
(d)	full model	63.4%	**66.9%**
(e)	no transformations	60.4%	64.4%
(f)	unsupervised STF	59.5%	64.2%
(g)	weakly supervised STF	61.6%	64.6%

Table 4.3: *Comparative segmentation results on MSRC.*

There are several conclusions to draw. (1) In all cases the ILP improves results. (2) The hierarchy of clusters in the STF gives a noticeable improvement. (3) The region priors alone perform remarkably well. Comparing to the segmentation result using only the STF leaf distributions (34.5%) this shows the power of the localized BOSTs that exploit semantic context. (4) Each aspect of the BOST adds to the model. While, without the ILP, score (b) is slightly better than the full model (d), adding in the ILP shows how the region priors and textons work together.[1] (5) Random transformations of the training images improve performance by adding invariance. (6) Performance increases with more supervision, but even unsuper-

[1]This effect may be due to segmentation forest (b) being over-confident: looking at the 5 most likely classes inferred for each pixel, (b) achieves 87.6% while (d) achieves a better 88.0%.

vised STFs allow good segmentations.

Given this insight, we compare against [92] and [103]. We use the same train/test split as [92] (though not [103]). The results are summarized in Figure 4.7, with further segmentation results in Figure 4.8. Across the whole challenging dataset, using the full model with ILP achieved a class average performance of 66.9%, a significant improvement on both the 57.7% of [92] and the 64% of [103]. The global accuracy also improves slightly on [92]. The image-level prior improves performance for all but three classes, but even without it, results are still highly competitive with other methods. Our use of balanced training has resulted in more consistent performance across classes, and significant improvements for certain difficult classes: cow, sheep, bird, chair, and cat. We do not use a Markov or conditional random field, which would likely further improve our performance [92].

These results used our novel learned and extremely fast STFs, without needing any slow hand-designed filter-banks or descriptors. Extracting the semantic textons at every pixel takes an average of only 275 milliseconds per image, categorization takes 190 ms, and evaluating the segmentation forest only 140 ms. For comparison [92] took over 6 seconds per test image, and [103] took an average of over 2 seconds per image for feature extraction and between 0.3 to 2 seconds for estimating the segmentation. Our algorithm is well over 5 times faster *and* improves quantitative results. We strongly believe a real-time implementation of the STFs is possible on a standard PC as the complexity is just $O(TD)$ per pixel.

The following parameter settings were used: $T = 50$ trees of depth $D = 14$, with training examples taken every 10 pixels in a random 50% of the training images, and using 1000 random feature tests and 20 random threshold tests at each split node. The ILP weighting term $\alpha = 0.5$. To add invariance, the training set was augmented: the original plus 3 copies with random transformations of rotation up to 6°, scaling up to 1.2x, left-right flipping, and affine intensity changes up to $1.2I + 0.05$. Training took under 2 hours for the segmentation forest.

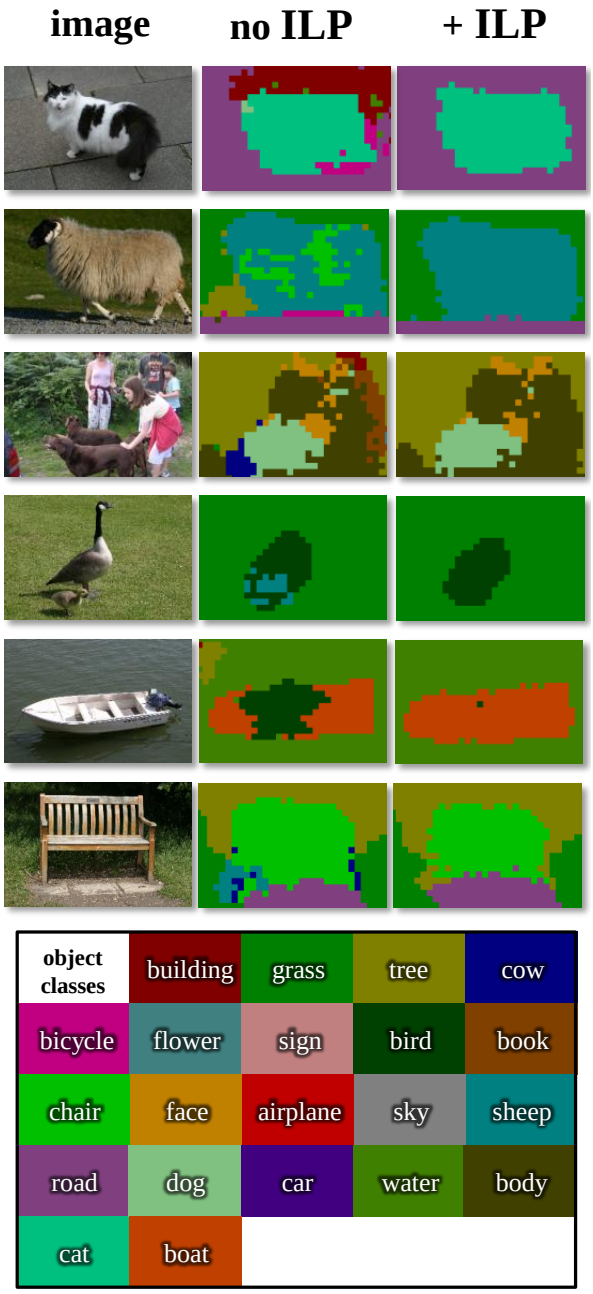
The resulting forest used a total of 24805 node features and 107311 region prior features.

4.4.2 Scenes Dataset

In addition, we show the results of our semantic segmentation algorithm on the Scenes dataset. This dataset was created automatically from anonymous user region annotations for the spatial envelope images of Oliva and Torralba [76] obtained from the LabelMe project [85]. We present it here to show that our algorithm can learn from noisy, automatically obtained label data and still achieve good segmentation performance on a large selection of images. We show segmentation accuracy and sample images in Figure 4.9. The forest used was depth 14, with 10 trees, trained on only the region priors.

4.5 Summary

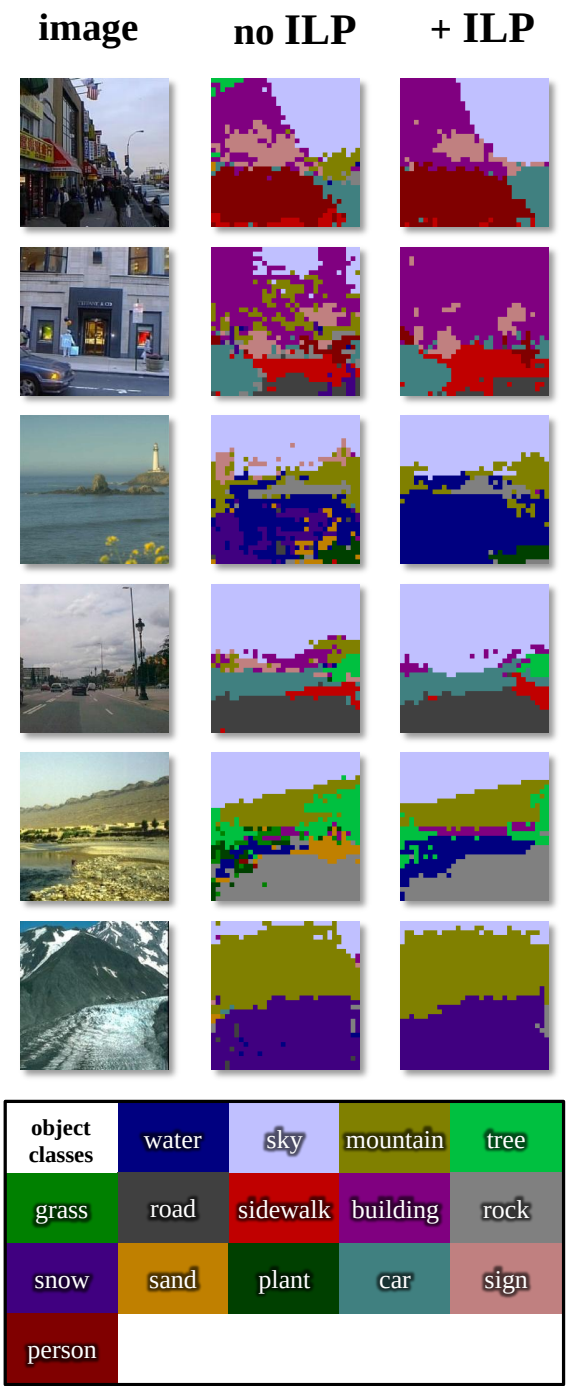
While this method of semantic segmentation gives excellent results, it requires ground truth labeled data in order to do so. While future research will determine the limits of the technique more accurately, the requirement of such data combined with its general scarcity places a practical limit on its application to datasets with larger numbers of categories. Additionally, as mentioned in Section 4.2.2, this algorithm achieves these results with such efficiency at the cost of a generative model which could potentially model the data better. Those limitations aside, these exciting results place in reach the goal of the automatic extraction of pixel-level semantic content, opening up new avenues for image search and retrieval which we shall explore in the rest of this thesis.



	[92]	[103]	Ours	Ours + ILP
building	**62**	52	41	49
grass	**98**	87	84	88
tree	**86**	68	75	79
cow	58	73	89	**97**
sheep	50	84	93	**97**
sky	83	**94**	79	78
airplane	60	**88**	86	82
water	53	**73**	47	54
face	74	70	**87**	**87**
car	63	68	65	**74**
bicycle	**75**	74	72	72
flower	63	**89**	61	74
sign	35	33	**36**	**36**
bird	19	19	**26**	24
book	92	78	91	**93**
chair	15	34	50	**51**
road	86	**89**	70	78
cat	54	46	72	**75**
dog	19	**49**	31	35
body	62	54	61	**66**
boat	7	**31**	14	18
Overall	71	-	68	**72**
Average	58	64	63	**67**

Figure 4.7: *MSRC21 segmentation results.* **Left:** Segmentations on test images using semantic texton forests. Note how the good but somewhat noisy segmentations are cleaned up using our image-level prior (ILP) that emphasizes the categories likely to be present. (Note we do not use a Markov or conditional random field which could clean up the segmentations to precisely follow image edges [92]). **Right:** Segmentation accuracies (percent) over the whole dataset, without and with the ILP. Our new highly efficient semantic textons achieve a significant improvement on previous work.

Figure 4.8: *Further MSRC segmentation results.*



	Ours	Ours + ILP
water	31	48
sky	76	84
mountain	24	23
tree	46	51
grass	64	69
road	46	62
sidewalk	39	29
building	39	52
rock	21	17
snow	53	49
sand	32	38
plant	22	17
car	48	47
sign	17	14
person	25	12
Overall	48	56
Average	39	41

Figure 4.9: *Scenes segmentation results.* **Left:** Segmentations on test images using semantic texton forests. **Right:** Segmentation accuracies (percent) over the whole dataset, without and with the ILP.

CHAPTER 5

IMAGE ANNOTATION AND LABELING

Image annotation is the task of automatically assigning a set of text labels to an image based on its visual content. A successful image auto-annotator can act as a trusted annotator for a text-based image retrieval algorithm, or simply categorize images by general topic for easy browsing. In Chapter 4, we discussed how to use semantic texton forests and a new model for image generation to create a semantic segmentation of an image, but the segmentation system we built stopped short of this model, instead sacrificing expressive power for speed and performance. In this chapter we discuss an alternative route to semantic segmentation which performs inference on the model to choose a topic for an image (annotation), and then uses the inferred topic to perform semantic segmentation (labeling).

5.1 Image Labeling

The cell model from Chapter 4 is similar in character to many algorithms and techniques from the machine translation community. A related model for the generation of text from topics was adapted for use in computer vision by Barnard *et al.* [2; 26] and ourselves [4], and is effective at a variety of vision tasks. The simplest version of this model was inspired by [44], though a full range of models is evaluated in [2] with recent work focusing on latent Dirichlet allocations and probabilistic latent semantic analysis [93; 9].

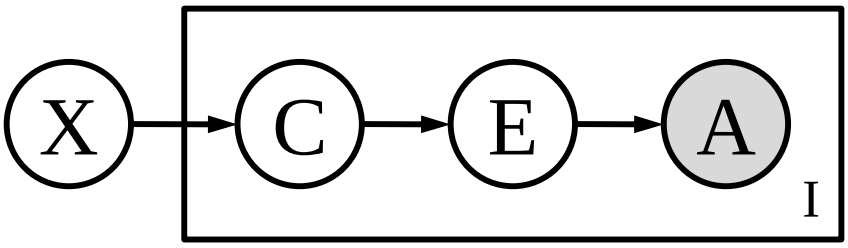


Figure 5.1: *Extended Cell Model*. We see here a graphical representation [8] of an extension to the cell model introduced in Chapter 4. We have added an additional random variable E which is generated from the cell category C, in turn generated from the image topic X. E has as its domain a set of known appearance models, *e.g.* a collection of image patches. These patches are held to generate the observed appearance A.

The focus of previous work has been on the semi-supervised (and in some cases, unsupervised) clustering of images into topics. The purpose of this was to subsequently use those topics to bridge the semantic gap in order to automatically provide text labels for those images. Our aim is slightly different. We wish to infer categories at the level of an image cell. In order to do this, we model the topic as a set of text labels, and perform inference over those sets. We are then able to determine the label likelihood of a word directly from $P(X)$. We are further able to use this to infer the image cell labels, performing a semantic segmentation and image annotation. The advantage of this model is that it is able to incorporate whatever data is known. In our case, we know the labels at each cell in the training data and can use this to aid in inference, however the model is fully capable of dealing with C being unobserved or partially observed without having to change the manner in which it is fit and in which inference is performed.

Before we proceed, the cell model we introduced in Chapter 4 must be extended. In its initial form, it generates pixel appearance directly from the category labels. In practice, the semantic texton forest adds a level of indirection to this process. In Figure 5.1 we see an

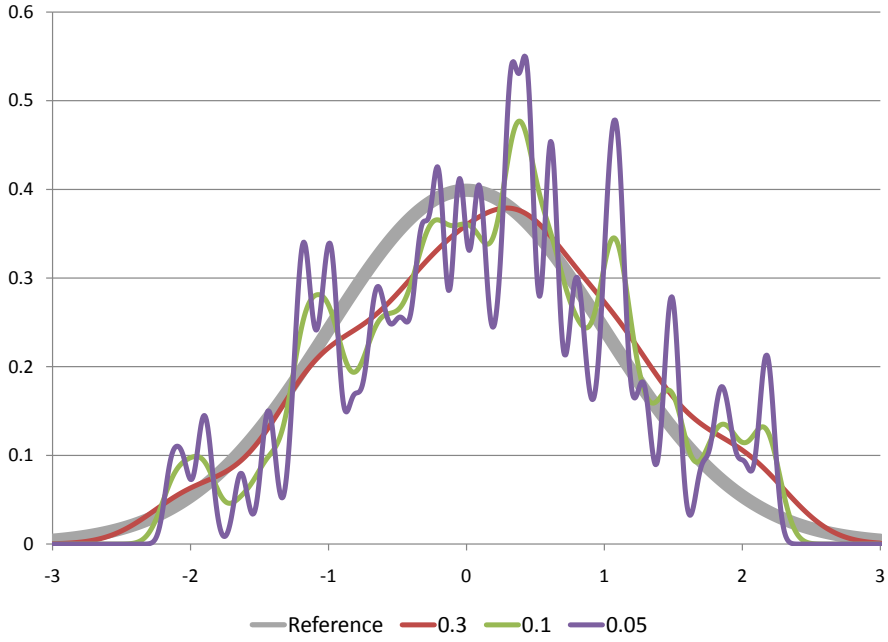


Figure 5.2: *Kernel Density Estimation* Here we have a Gaussian distribution, represented by the gray curve, estimated by 100 samples drawn from it using different bandwidths. As bandwidth (equivalent to standard deviation in this case) decreases, the noise of the estimation increases.

extension to the original cell model which includes a sample point E, which generates the appearance through the addition of some noise. E has as its values the leaf nodes in the semantic texton forest, and each pixel is assumed to have been produced by a generative process from that node. Thus $P(C|A)$ as described previously is in fact $P(C|E)$, as it is associated with all pixels which reached the leaf node associated with a value of E regardless of their individually different appearances, which are assumed to have been caused by added noise as part of a generative process.

The graphical model we have shown is an example of a *kernel density estimator* [8]. A kernel density estimator uses each training data point individually to model a complex probability surface. To give an example, suppose that we have been given a set $Q = \{q_i | i = 1, \ldots, N\}$ of points drawn from an unknown distribution. If we have some knowledge about the nature of this unknown distribution, say that it was a normal distribution, we can es-

timate its value at an unseen point r as the average of the responses of kernel functions centered around the points in Q:

$$P(r) = \frac{1}{Nh} \sum_i K\left(\frac{r - q_i}{h}\right) \tag{5.1}$$

where K in this case would be the standard normal distribution:

$$K(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2} \tag{5.2}$$

and h is the *bandwidth* of the estimator. In the case where K is a standard normal distribution, h is equivalent to the standard deviation. An example of the result for different values of h can be seen in Figure 5.2. In the case of a semantic texton forest, every image pixel is assigned to a number of leaf nodes equal to the number of trees in the forest. These assignments act as a binary kernel function K, in which K is 1 for all leaf nodes to which a pixel belongs and zero otherwise.

Let us now look in detail at the model in Figure 5.1. The possible values of the random variables are:

$$X \in \{\{w_k\} | w_k \in W\} \tag{5.3}$$

$$A \in \Re^N \tag{5.4}$$

$$E \in \{e\} \tag{5.5}$$

$$C \in \{c | c \in W\} \tag{5.6}$$

where N is the number of channels in the image representation, e is a leaf node in the forest and W is the vocabulary of words. We will denote an arbitrarily sized set of words in the vocabulary W as $W_k = \{w_k\}$ to simplify notation from now on.

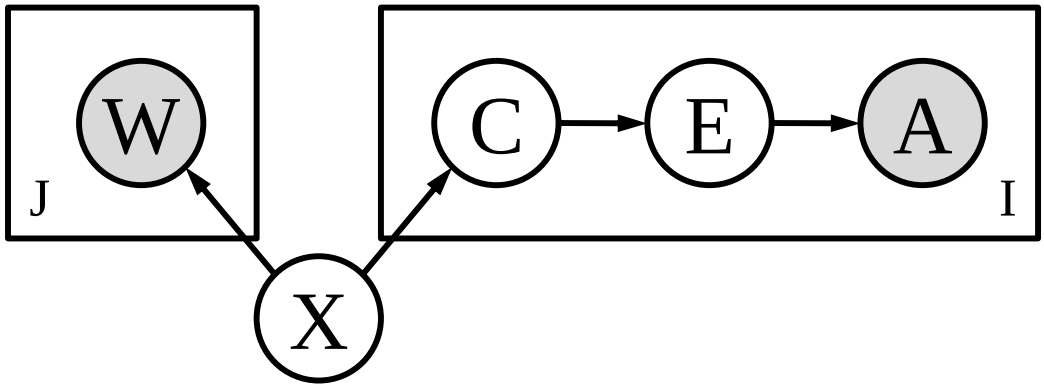


Figure 5.3: *Combined Word/Appearance Model*. This model is very much like the separated word/appearance model of Barnard *et al*. [2], though modified to use a kernel density estimator to model appearance. In this graph the random variables are as in Figure 5.1, with the addition of the random variable W, which has as its domain one of a number of words in a large image label dictionary. Here, the topic X links the appearance and the labels, which are presumed to be conditionally independent.

In Figure 5.1 we have shaded in only A, as this is the only variable which is observed during inference. Our training data consists of the pixels themselves A, their category labels C and the image label sets X. These are used to train a semantic texton forest, which associates a learned distribution $P(C|E)$ with every leaf node e. We use the forest to act as the binary kernel function K in order that we can marginalize over E during inference. We model the topic as a set of words in the dictionary, and the cell category labels as being drawn from the same dictionary. Thus, the choice of $P(C|X)$ is very easily chosen as a uniform prior over the words in X. One of the advantages of this model is that the domain of X and the choice of $P(C|X)$ can be chosen in different ways such that the label vocabulary is different, and potentially broader, than the segmentation vocabulary. Alternatively, the topics can be learned automatically using expectation maximization [25]. While we do not do so in this chapter, it is possible to use the topic to link the appearance of the image to a set of labels as is done in [2] using a model such as that shown in Figure 5.3. In this scenario, the labels for an image are decoupled from appearance entirely, joined only by shared topics which generate appearance and labels independently. In our case, we use $P(C|X)$ to find

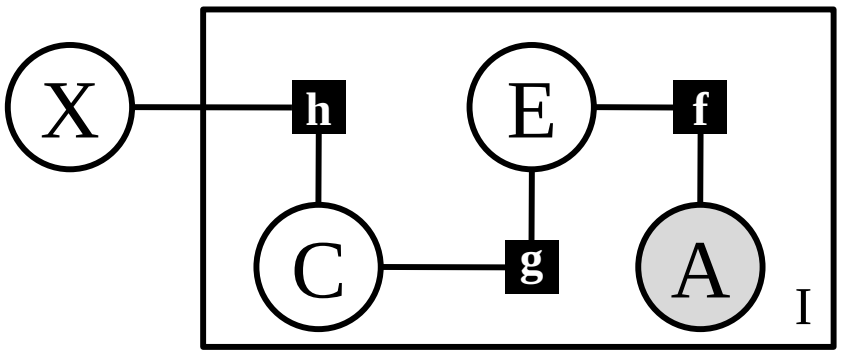


Figure 5.4: *Factor Graph*. This is a factor graph [35; 58] expansion of the model in Figure 5.1. The lower case letters f,g and h are the factors of the model and encode the ways in which the random variables are related.

the $P(X|C)$ using Bayes' rule as follows:

$$P(X|C) = \frac{P(C|X)P(X)}{\sum_X P(C|X)P(X)} \tag{5.7}$$

where $P(X)$ is based upon the occurrence of the topic X in the training data, or a learned mixing parameter in the case when topics are learned automatically.

5.2 Inference

The sum-product algorithm provides a mechanism by which to compute any marginal in the graphical model efficiently. In order to apply it, we must first transform the model into a factor graph, as shown in Figure 5.4[35; 58]. Using X as the root of the graph, incoming

messages from the leaves are as follows:

$$\mu_{A_i \to f_i}(A_i) = 1 \tag{5.8}$$

$$\mu_{f_i \to E_i}(E_i) = f(A_i, E_i) \tag{5.9}$$

$$\mu_{E_i \to g_i}(E_i) = \mu_{f_i \to E_i}(E_i) \tag{5.10}$$

$$\mu_{g_i \to C_i}(C_i) = \sum_{E_i} g(C_i, E_i) \mu_{E_i \to g_i}(E_i) \tag{5.11}$$

$$\mu_{C_i \to h_i}(C_i) = \mu_{g_i \to C_i}(C_i) \tag{5.12}$$

$$\mu_{h_i \to X}(X) = \sum_{C_i} h(X, C_i) \mu_{C_i \to h_i}(C_i) \tag{5.13}$$

$$\tag{5.14}$$

given the appearance is observed, and messages going back out from the root to the category nodes are:

$$\mu_{X \to h_i} = \prod_{j \neq i} \mu_{h_j \to X}(X) \tag{5.15}$$

$$\mu_{h_i \to C_i} = \sum_{X} h(C_i, X) \mu_{X \to h_i}(X) \tag{5.16}$$

given the topic is observed. The factors are calculated as:

$$f(X, Y) = P(X|Y) \tag{5.17}$$

for all random variables X and Y. Using these equations, we find that the marginal likelihood of a particular topic is

$$P(X) = \prod_i \sum_{C_i} P(X|C_i) \sum_{E_i} P(C_i|E_i) P(E_i|a_i) \tag{5.18}$$

where a_i is the observed appearance at image cell i. We can now use this marginal to infer the value of C_i:

$$P(C_i) \propto \sum_X P(C_i|X)P(X) \tag{5.19}$$

The inference of X gives us an annotation of the image, and the inference of C_i gives us a semantic segmentation.

5.3 Experiments

We performed experiments with annotation and labeling to test the effectiveness of this model. We first infer the value of X as the maximum *a posteriori* value as shown in Equation 5.18. These annotations of the images are compared against the ground truth annotations, and used to compute PR and ROC curves. We then label each image cell using $P(X)$, performing a semantic segmentation.

5.3.1 Annotation

We performed experiments on both datasets, and show the AuC (Area under the ROC Curve) and AP (Average Precision) for all words. Images are ranked by the *label likelihood* of a word w, which is calculated as

$$\sum_{W_k|w \in W_k} P(X = W_k) \tag{5.20}$$

The set of words in the label vocabulary for the MSRC dataset is equivalent to the possible pixel categories, and a label is in the label set of an image if it occurs in the ground truth segmentation of the image. For the scenes dataset, the topic corresponds to the scene label of the image, and so $P(C|X)$ is a learned distribution from the training data (see Appendix B for details) where the domain of C is the pixel labels. The labeling performance results

Category	AP	AuC
building	0.418	0.725
grass	0.913	0.945
tree	0.517	0.797
cow	0.795	0.978
sheep	0.922	0.990
sky	0.855	0.921
airplane	0.826	0.989
water	0.647	0.888
face	0.749	0.924
car	0.440	0.899
bicycle	0.395	0.941
flower	0.735	0.965
sign	0.204	0.816
bird	0.310	0.766
book	0.821	0.986
chair	0.278	0.913
road	0.627	0.864
cat	0.490	0.940
dog	0.338	0.820
body	0.707	0.905
boat	0.613	0.927
Mean	**0.600**	**0.900**

MSRC21 Annotation Performance.

Category	AP	AuC
water	0.372	0.658
sky	0.948	0.798
mountain	0.559	0.682
tree	0.700	0.699
grass	0.201	0.799
road	0.646	0.807
sidewalk	0.570	0.862
building	0.720	0.717
rock	0.118	0.656
snow	0.124	0.769
sand	0.145	0.759
plant	0.245	0.680
car	0.566	0.820
sign	0.241	0.776
person	0.272	0.729
Mean	**0.428**	**0.747**

Scenes Annotation Performance.

Table 5.1: *Annotation Performance.* Annotation performance is measured using average precision and area under the curve, where images are ranked by the label likelihood of a word in the vocabulary. Correct images are those whose label set contains the label.

for the MSRC21 dataset are shown in Figure 5.5. Similarly, the results for the Scene dataset are shown in Figure 5.6. The labels shown are the maximum *a posteriori* value of X for the image. Annotation performance in term of label likelihood is shown in Table 5.1. As can be seen, the system is able to infer the topic of an image to a good level of accuracy.

5.3.2 Labeling

For the labeling case, we divide a test image into cells in the same way as in Chapter 4 and infer $P(C|E)$ for each cell. We use these to compute the distribution $P(X)$ using Equation 5.18. We then infer the label of C using $P(C|E)$ and $P(C|X)$, as in Equation 5.19. Sample segmentations and accuracies for the MSRC21 and Scenes datasets can be observed in

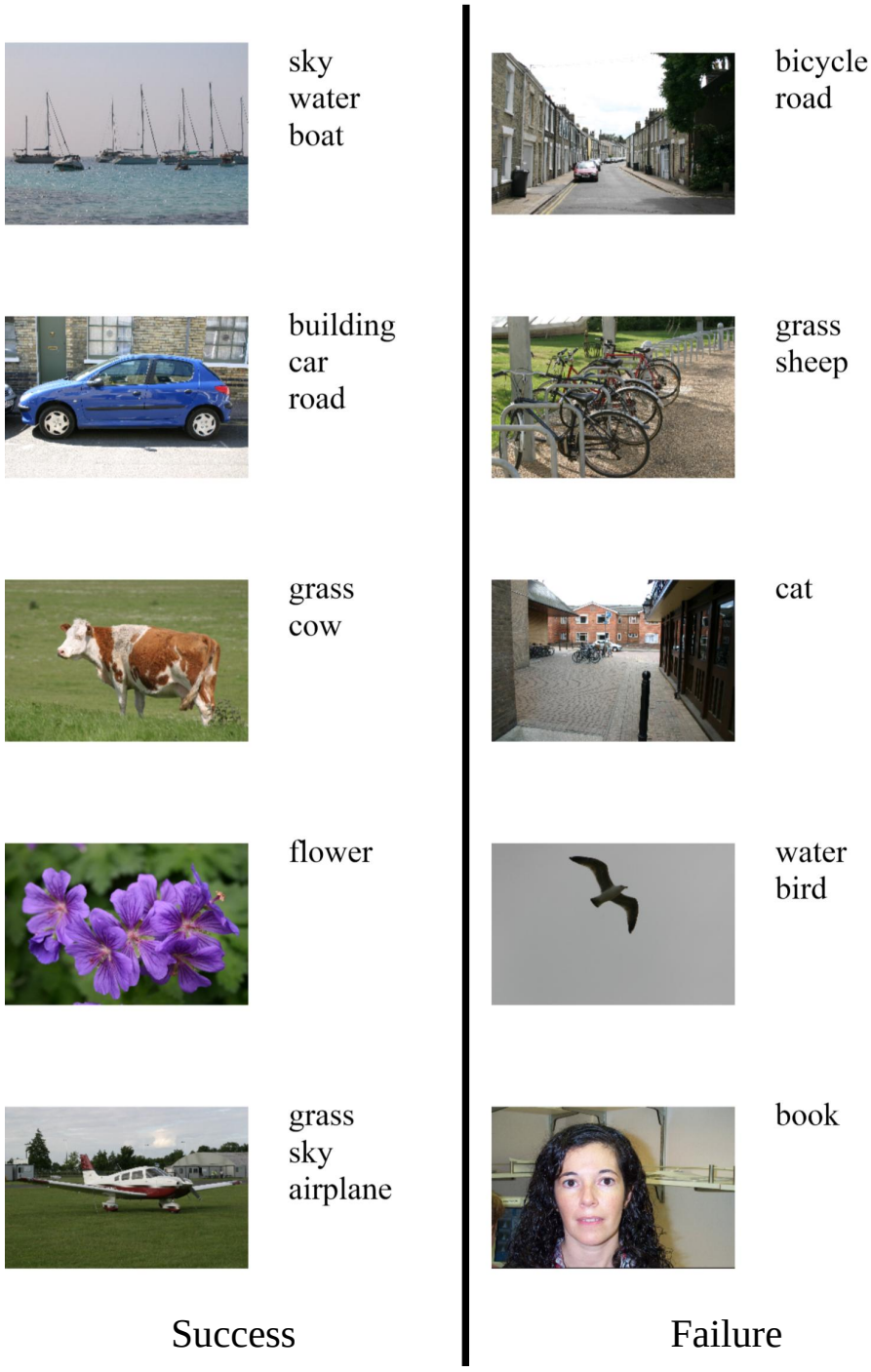


Figure 5.5: *MSRC21 Annotations*. The annotations on the left are correct, the annotations on the right are partially or wholly incorrect, and represent standard failure cases. In these failure cases, the correct segmentation and label likelihood can still be achieved due to marginalizing over X, even though the maximum posterior likelihood X value is incorrect.

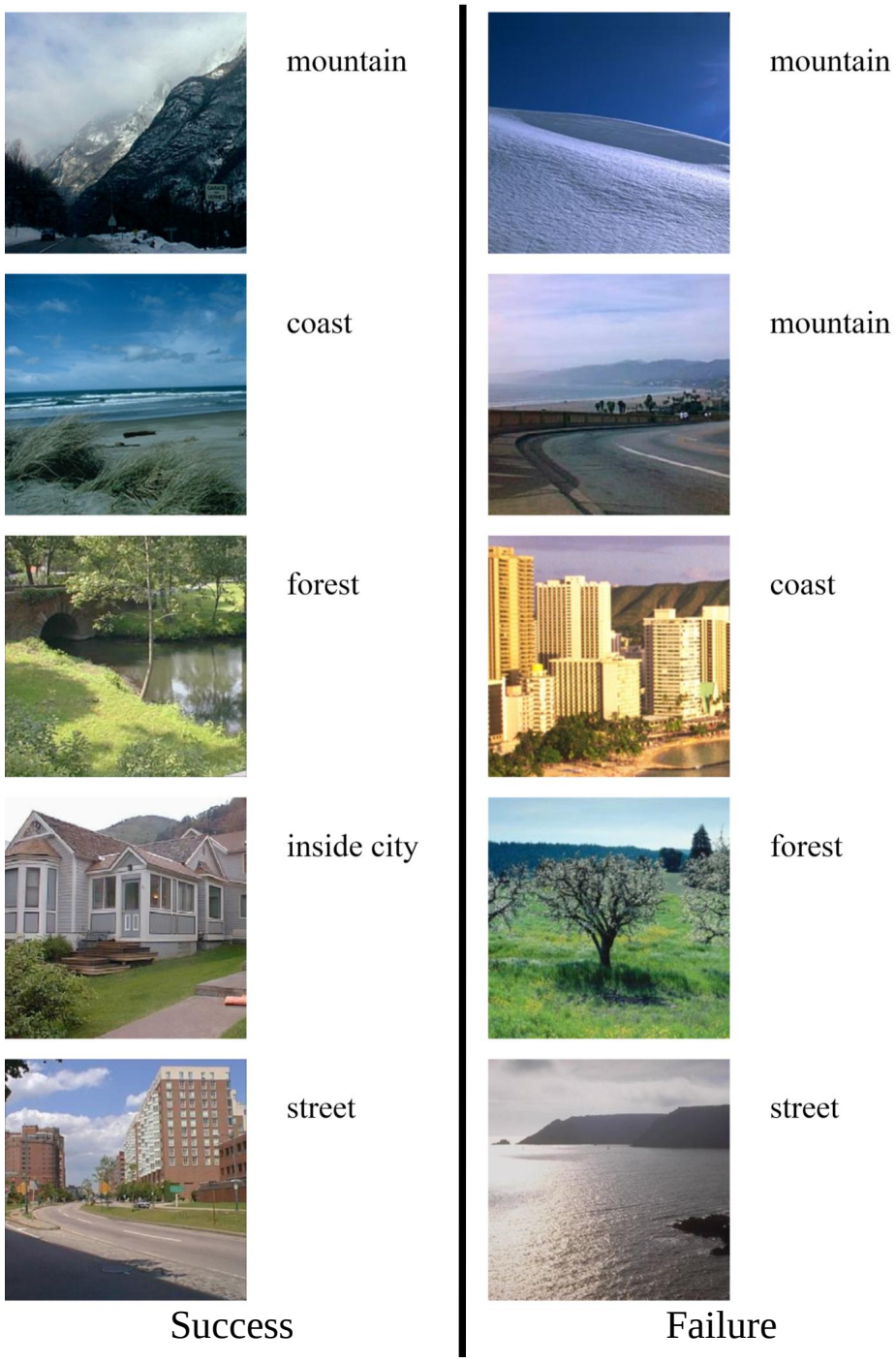
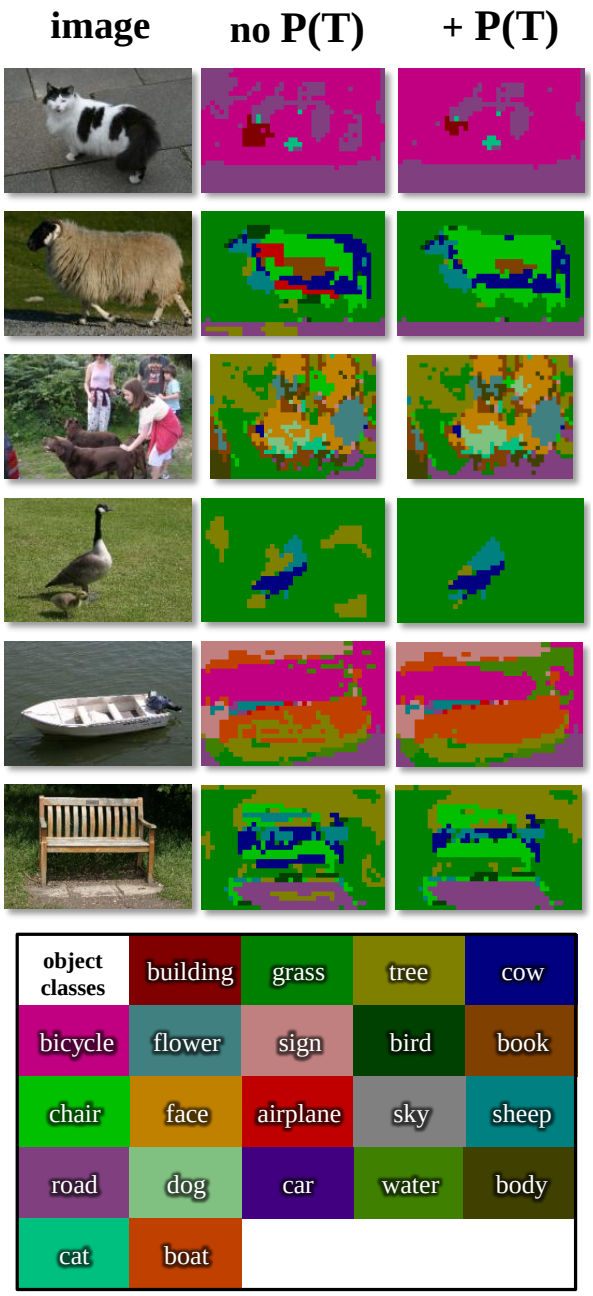


Figure 5.6: *Scenes Annotations*. The annotations on the left are correct, the annotations on the right are incorrect, and represent standard failure cases. In general, when the system makes a mistake it is understandable, and usually due to ambiguity in the image. Sometimes, however, it is clearly wrong, as with the last image.

Figures 5.7 and 5.8, respectively. The use of the inferred topic X improves segmentation performance significantly, as shown in the figures.

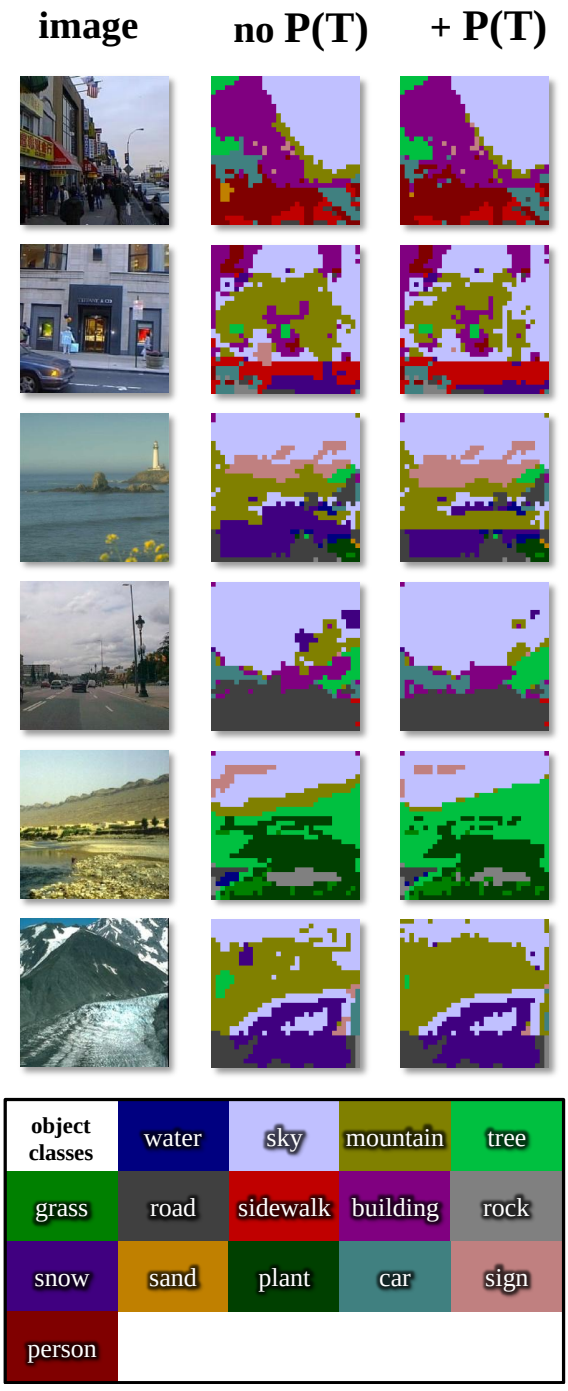
5.4 Summary

We have explored a fully generative alternative to the semantic segmentation algorithm in Chapter 4. While it does not achieve the same level of performance as that algorithm, it should be noted that its performance is achieved in the absence of contextual information (there is no second-level forest) and as a generative model it is more able to adapt to situations with more pixel categories. The introduction of a conditional random field [59] modification to the model would likely improve results considerably, given a similar approach in [92]. We have also provided a formulation of a semantic texton forest as the kernel function of a kernel density estimator, which places the framework on firm theoretical footing. Using a generative model additionally opens up the possibility of training the entire model in the absence of pixel-level ground truth data, as model fitting can now be performed by variational methods [8]. This chapter ends our discussion of semantic segmentation, and we now turn to its application to image search and retrieval. We will first introduce the concept of semantic composition in Chapter 6, and then the Palette Search system in Chapter 7.



	STF	STF + T
building	11	6
grass	82	93
tree	46	38
cow	63	67
sheep	67	71
sky	72	70
airplane	75	79
water	25	33
face	77	80
car	30	30
bicycle	49	57
flower	55	64
sign	12	15
bird	14	19
book	50	77
chair	24	35
road	63	69
cat	27	33
dog	27	17
body	42	41
boat	19	20
Overall	51	55
Average	44	48

Figure 5.7: *MSRC21 labeling results.* **Left:** Labelings on test images using semantic texton forests. **Right:** Segmentation accuracies (percentage) over the whole dataset, without and with topic inference.



	STF	STF + T
water	13	9
sky	73	84
mountain	18	21
tree	35	51
grass	62	56
road	38	57
sidewalk	46	23
building	31	41
rock	11	11
snow	54	43
sand	17	12
plant	21	12
car	40	36
sign	20	8
person	29	20
Overall	41	48
Average	34	32

Figure 5.8: *Scenes segmentation results.* **Left:** Segmentations on test images using semantic texton forests. **Right:** Segmentation accuracies (percentage) over the whole dataset, without and with topic inference.

CHAPTER 6

SEMANTIC COMPOSITION

The problem of image query representation is central to the problem of image search and retrieval. The user has an image in their mind which they wish to find. Communicating the information about that image to a search program is therefore the first task which any image search program must achieve. Eakins [27] gives a helpful delineation of image search into three levels, ranging from the very concrete (*e.g.* images with lots of green) to the highly abstract (*e.g.* images depicting "joy"). The first level comprises retrieval by what Gudivada and Raghavan [38] describe as primitive features such as color, texture, shape or the spatial location of image elements. Eakins divides this level further into the following sub-categories:

- retrieval by color feature (find all pictures containing yellow and blue regions)
- retrieval by texture (find images containing regions with texture similar to that of woven cloth)
- retrieval by shape (find drawings similar in shape to this sketch)
- retrieval by spatial location (find pictures with objects of interest in the top left-hand corner)
- retrieval by a combination of the above (find images containing yellow stars arranged in a ring)

which is similar to image retrieval using the basic image cues described in Chapter 2.

The third level is the retrieval of images by way of highly abstract concepts and ideas (*e.g.* "Find a photograph of Morris dancing", "Find a painting of a forlorn lover"). While this is still out of reach both in terms of query representation and image understanding, we present in this chapter a method of query representation and image retrieval which achieves the second level of image search. This level consists of retrieval by derived attributes (Gudivada and Raghavan describe these as logical features), involving some degree of logical inference about the identity of the objects depicted in the image. This is referred to as retrieval by *semantic content*, and corresponds to Panofsky's *pre-iconographic* level of picture description [78].

Most search programs which are successful with this kind of search employ the use of text describing the image. Text-based methods allow the user to specify the semantic content of the image, *i.e.* the words which represent the objects in the scene that the user is envisioning. The major drawback to this is the ideal requirement of trusted labels for all images in the dataset, though we touched on a possible solution to this problem in Chapter 5. Even a perfect text based system, however, cannot capture the compositional information presented by the content based image retrieval methods which utilize image statistics and edge sketches. In this chapter, we further develop the concept of *semantic composition*, which we first presented in a primitive form in [53]. Semantic composition is unique in that it provides a framework within which queries can be specified using both semantic content (what) and composition (where).

6.1 Bayesian Image Retrieval

When a user is searching for an image, he has a mental prototype of that image which he will match against results. The "correct" result, in effect the result which an automatic

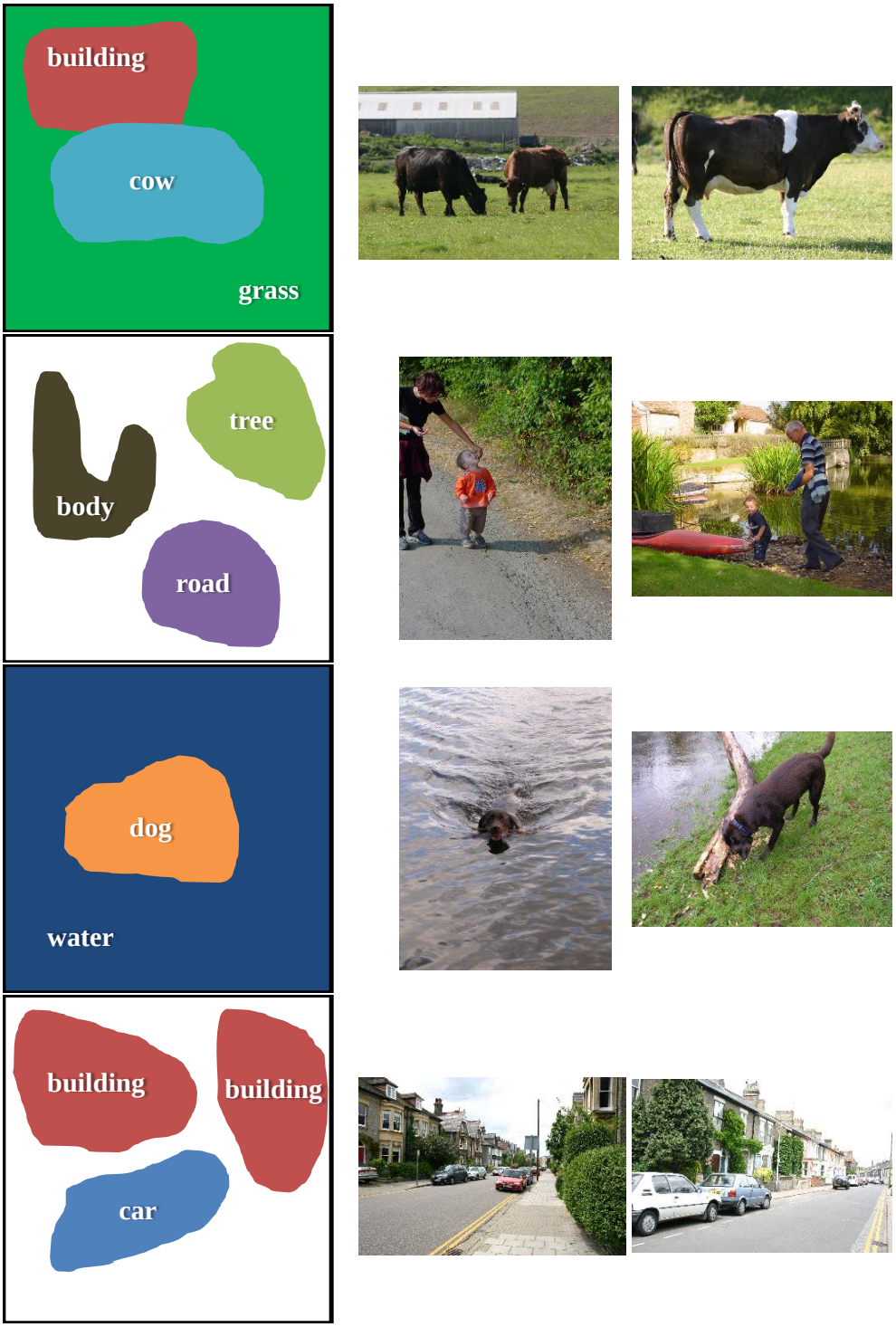


Figure 6.1: *Examples of Semantic Composition* Here are several examples of semantic compositions on the left and possible images which satisfy them. We see here partial and approximate matches as well as exact matches. A semantic composition is a specification of semantic labels for pixels in the target image, and matching images are those which adhere to those labels to some degree. An exact match is an image which adheres to all of the labels in the query, whereas partial matches may adhere to a subset of the categories, or only portions of the labeled regions.

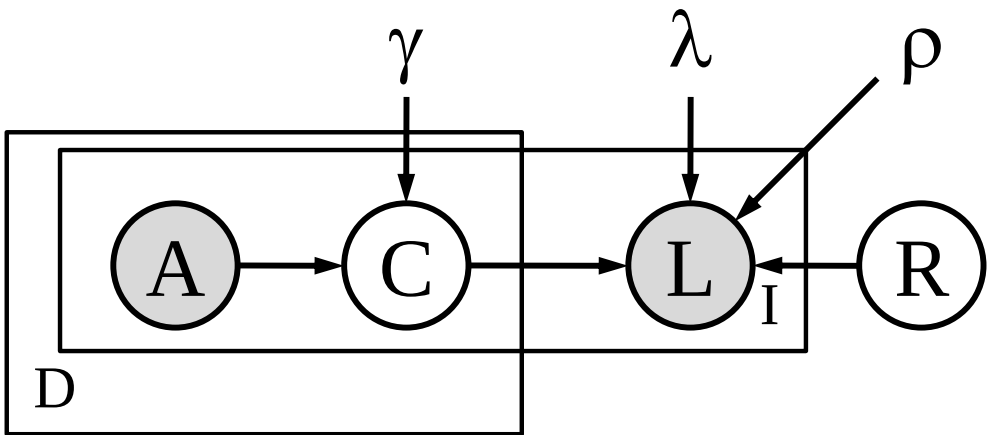


Figure 6.2: *Graphical Model*. The model is based on the idea that there is an image in the database which the user is trying to find. Their idea of that image has caused them to create a set of observed labels L set out on a regular grid indexed by i, with size I. The variable R is the image which the user is looking for, and A, C are associated with images in the database, of which there are D. We want to find the image in the database which best explains the labels the user has provided. γ, λ and ρ are the parameters of the model. See text for details.

image retrieval system should rank highly, is that which most closely matches that mental prototype. The problem of representing this prototype in a manner which is meaningful to the search system is the heart of the problem of query representation. We propose that this mental prototype consists of a *semantic composition*, that is to say a spatial arrangement of semantically contiguous regions (*i.e.* self-contained regions in which all pixels are produced by the same independent entity). Some examples of semantic compositions and possible corresponding images can be seen in Figure 6.1.

The semantic composition provides a probabilistic framework within which the user can specify their ideal image. As can be seen in Figure 6.2, the structure is based upon the model for semantic segmentation introduced in Chapter 4, in that it is composed of a grid of independent cells that divide the image and which have associated with them an appearance $A \in \{a|a = \Re^N\}$ and a category label $C \in \{c\}$. However, in this model there is the query image, consisting of label variables L, and the reference images in the database. The image which the user is looking for is represented in the model by the random variable

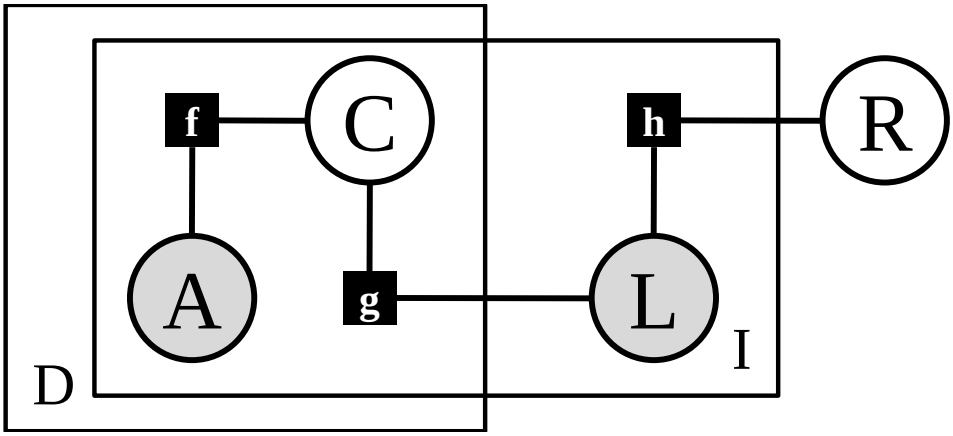


Figure 6.3: *Factor Graph* This is a factor graph expansion of the semantic composition model to aid in the application of the sum-product algorithm in the text [35; 58].

$R \in \{d|0, \ldots, D\}$. The query labels $L \in \{w\}$ are provided by the user, and reference image node values are precomputed and stored in the database. As can be seen, the appearance of the reference images and the labels of the query images are observed. Provided that we are able to compute $P(C_i|A_i)$ and $P(L_i|C_i)$ the system is able to infer how well a reference image in the database corresponds to the query which the user has presented.

In effect, the semantic composition is a semantic segmentation as discussed in Chapter 4 which is produced *ex nihilo* and then mapped onto potential results in what essentially amounts to a maximum likelihood decision: what is the maximally likely image to have produced this segmentation? The reason the model is called the semantic composition is because it represents two distinct quantities: pixel meaning (semantics, represented by the value of L in the model), and where and in what proportion that meaning is distributed within the image (composition, represented by the subscript i).

6.2 Efficient Compositional Search

The model shown in Figure 6.2 enables us to infer $P(R)$ by marginalizing over the other variables. In order to find an efficient manner in which to compute this, we must first con-

vert the graphical model into a factor graph, as can be seen in Figure 6.3. We can use this factor graph to highlight exactly which computations need to be performed to compute this marginal in the most efficient way through the use of the sum/product algorithm [35; 58]. If we choose $P(R)$ as the root we pass the following messages from the leaf nodes:

$$\mu_{A_{di} \to f_{di}}(A_{di}) = 1 \tag{6.1}$$

$$\mu_{f_{di} \to C_{di}}(C_{di}) = \sum_{A_{di}} f(C_{di}, A_{di}) \mu_{A_{di} \to f_{di}}(A_{di}) \tag{6.2}$$

$$\mu_{C_{di} \to g_{di}}(C_{di}) = \mu_{f_{di} \to C_{di}}(C_{di}) \tag{6.3}$$

$$\mu_{g_{di} \to L_i}(L_i) = \sum_{C_{di}} g(C_{di}, L_{qi}) \mu_{C_{di} \to g_{di}}(C_{di}) \tag{6.4}$$

$$\mu_{L_i \to h_i}(L_i) = \prod_{d} \mu_{g_{di} \to L_i} \tag{6.5}$$

$$\mu_{h_i \to R}(R) = \sum_{L_i} h(L_i, R) \mu_{L_i \to h_i}(L_i) \tag{6.6}$$

and the following messages back out to the leaves:

$$\mu_{R \to h_i}(R) = \prod_{j \neq i} \mu_{h_j \to R}(R) \tag{6.7}$$

$$\mu_{h_i \to L_i}(L_i) = \sum_{R} h(R, L_i) \mu_{R \to h_i}(R) \tag{6.8}$$

$$\mu_{L_i \to g_{di}}(L_i) = \mu_{h_i \to L_i} \prod_{e \neq d} \mu_{e_i \to L_i} \tag{6.9}$$

$$\mu_{g_{di} \to C_{di}}(C_{di}) = \sum_{L_i} g(C_{di}, L_i) \mu_{L_i \to g_{di}} \tag{6.10}$$

$$\mu_{C_{di} \to f_{di}}(C_{di}) = \mu_{g_{di} \to C_{di}}(C_{di}) \tag{6.11}$$

$$\mu_{f_{di} \to A_{di}}(A_{di}) = \sum_{C_{di}} f(C_{di}, A_{di}) \mu_{C_{di} \to f_{di}} \tag{6.12}$$

where the factors are computed as follows:

$$f(C_{di} = c, A_{di}) = P(C_{di} = c|A_{di}) = \gamma[d, i, c] \tag{6.13}$$

$$g(C_{di} = c, L_i = w) = P(L_i = w|C_{di} = c) = \lambda[c, w] \tag{6.14}$$

$$h(L_i = w, R = d) = \rho[d, w] \tag{6.15}$$

where ρ is a learned distribution over user labels w for each database image d, λ is a distribution over user labels w for category c in the database, and γ is distribution over category c at location i in database image d, who initial value is produced by an automatic semantic segmentation algorithm such as the ones presented in Chapters 4 and 5 from the observed appearance A_{di}.

When performing a search, we want to calculate the marginal likelihood of $P(R = d)$ for all documents d. By arranging the messages we can find the most optimal way of computing this value. First, we look at all messages coming into R:

$$P(R = d) = \prod_i \mu_{h_{qi} \to R}(R = d) \ . \tag{6.16}$$

We then expand this out to the labels L_i in the image, using the observed values from the user $L_i = w_i$:

$$P(R = d) = \prod_i h(L_i = w_i, R = d)\mu_{h_i \to L_i}(L_i = w_i) \ . \tag{6.17}$$

This equation is written as though all L_i are observed. However, in practice this is not the case and indeed above we have written the messages for the unobserved case. Let $K = \{k \in I\}$ be the set of known grid cells and $U = \{u \in I\}$ be the set of unknown grid cells. The

correct equation (with observed random variables replaced by their values) is then:

$$P(d) = \prod_k h(w_k, d)\mu_{h_i \to L_i}(w_k) \prod_u \sum_{L_u} h(L_u, d)\mu_{h_i \to L_i}(L_u) \tag{6.18}$$

If we complete the expansion including replacing factors with their computations, we arrive at the marginal likelihood equation for $P(d)$:

$$P(d) = \prod_{k \in K} \left[\rho[d, w_k] \sum_c \lambda[c, w_k]\gamma[d, k, c] \right] \prod_{u \in U} \left[\sum_w \rho[d, w] \sum_c \lambda[c, w]\gamma[d, u, c] \right] \tag{6.19}$$

For unobserved nodes, while one option is to compute the sum over all values of L_u, another is to limit w to the distinct values of the known labels. Thus, the sum over values of L_u becomes

$$\sum_{w \in W_k} \rho[d, w] \sum_c \lambda[t, w]\gamma[d, u, c] \tag{6.20}$$

where $W_k = \{w_k | k \in K\}$, without repetition.

6.3 Summary

The practical outcome of this theoretical framework is the Palette Search image retrieval program. In the next chapter we will introduce and explore that implementation of this system, showing how the user interface corresponds to the various parts of the model, describing its features and showing its results.

CHAPTER 7

PALETTE SEARCH

In the last chapter we discussed the theory underlying semantic image search, which uses the semantic composition to form queries and as a probabilistic model for image comparison. To prove the efficacy of this method of image search and retrieval, we have developed an automatic image search and retrieval program called Palette Search. In this chapter we will briefly look at the history of image search and retrieval, and then move on to the Palette Search system, its design and its implementation.

7.1 Image Retrieval

There has been a significant body of previous research performed in the area of image search and retrieval. In general, there are two approaches: those which concentrate on words which correspond to the image through the use of manual or automatic captioning systems, and those which retrieve images based on photometric and geometric statistics. Text-based methods draw heavily from the text search and retrieval community, and as they rely mostly on associated captions as opposed to the visual characteristics of an image, they do not have much bearing on image retrieval *per se*. We refer the reader to [98] for a review of current text-based semantic search tools. We will, however, examine the state of the art in content based image retrieval, especially as pertains to the semantic understanding of images, and

then examine other techniques which use Bayesian inference for image retrieval.

7.1.1 Content Based Image Retrieval

The field of content based image retrieval has been very active, given the urgency created by the increasing number of searchable images and thus the demand for intuitive, effective image search systems. The early work in the field, excellently summarized by Smeulders *et al*. in [95], mostly concentrates on the description of images using the kinds of features discussed in Appendix A to create appearance models which can be used for retrieval. The classic example from this period is the *Query By Image Content* (QBIC) system of Flicker *et al*. [33], which was one of the first to perform large scale image searches using image content, and arguably the most successful. Of particular note is the work of Shaffrey [89], which used the dual-tree complex wavelet transform to describe automatically segmented image regions and allowed query by segment through large databases. This and other *Query By Visual Example* (QBVE) systems which grew out of QBIC [100] remain limited by the fact that they do not understand anything about the underlying causes of image appearance, but rather find images which are visually similar to the query specifications. This is referred to as the *semantic gap* between user and machine, in which the user becomes increasingly frustrated because he does not understand the results which his queries are producing. In the case of QBVE systems, they can be extended to incorporate semantics of a kind through the use of image labels. If a QBVE system has a database which contains images associated with words available to it, the system can use all of the images in the database which are associated with a user provided semantic label to query by example, with some success [16]. Even so, this is limited by the database, its captions, and the fact that there are some concepts which have highly variable appearance for which this technique is insufficient.

Query By Semantic Example (QBSE) systems create models of image concepts which cap-

ture the relationships between words and pictures [2; 26], enabling the ability to annotate images automatically with labels based upon their visual appearance. Given the nature of how these systems are trained, however, they require large quantities of data. Their strength lies in their ability to perform semantic inference, to connect the appearances of different images to their labels by way of global image topics. Thus, by learning an "ocean" topic, they are able to connect visual information from images which have labels such as "water", "sea", "ocean", "sky", "boat", and "ship" labels, and learn which words are synonymous and the fact that that they all tend to occur together. Our system is the first which uses *Query By Semantic Composition* (QBSC), whereby the user can search using both semantic concepts and their positions to return images which closely match the desired target. While Palette Search, the implementation of the system in this chapter, uses the semantic segmentation technique of Chapter 4, it is important to note that it is able to incorporate any techniques which are capable of inferring $P(C|A)$ and $P(L|R)$ and thus provides a framework within which this area can be explored in future research.

7.1.2 Bayesian Methods

There is a rich recent history in CBIR of methods which have utilized Bayesian statistics for image retrieval. Considerable research has been done on the subject by Vasconcelos *et al*. [100; 102] (a complete overview can be found in [101]). A recent system by Heller and Ghahramani [42] utilizes binary feature vector representations to perform augmented QBVE with a fully Bayesian model. The previously mentioned QBSE system of Barnard *et al*. [2] has not only been used in image annotation and retrieval but extended for use in cross-modal disambiguation [4; 5]. The QBVE system of Yavlinksy *et al*. [107] models image features directly from the words, eschewing the topic based model approach of Barnard *et al*.. Their work uses a non-parametric kernel density estimator based on the Earth Mover's

Distance [82], using Bayes rule to obtain $P(W|X)$ where X are image features and W is a word in the vocabulary.

One of the pioneering systems in the use of relevance feedback as an integral part of image retrieval is the PicHunter system of Cox *et al*. [20], which models uncertainty in the users' goal in order that it can optimally select sets of images for presentation to the user, with each user action removing uncertainty about that for which the user is looking. The innovative label-propagation framework of Zhu *et al*. [111] provides an attractive manner in which to perform semi-supervised learning which has been utilized as another way of incorporating relevance feedback [45]. Our work differs from all previous systems due to the fact that it has available to it a per-pixel category distribution for each reference image, thus opening up the ability to build a probabilistic model for image comparison which acts on the level of a grid cell. This model, which we presented in Chapter 6, allows us to compare images directly to the user's query, which incorporates both semantic category and position within the image.

7.2 Overview

In this chapter, we present the Palette Search image retrieval system. The system utilizes the semantic composition model to perform image retrieval by ranking images in the database by the marginal likelihood of $P(R = d)$ where d is an image in the database, computed as

$$P(d) = \prod_{k \in K} \left(\rho[d, w_k] \sum_c \lambda[c, w_k] \gamma[d, k, c] \right) \prod_{u \in U} \left(\sum_w \rho[d, w] \sum_c \lambda[c, w] \gamma[d, u, c] \right) . \quad (7.1)$$

Naturally, due to computational constraints this is computed and compared in log space. The retrieval system consists of the user interface, which we have adapted and updated from the semantic photo synthesis interface of [53], the image database, and the comparison

model. It enables the user to provide the observed values of L_i (whose indices form the set of known labels K) to the model. λ is a translation or word frequency table, which models how the labels the user can provide map onto the topics which form the domain of C. Palette search is optimized for use by users of stylus-based personal computers and devices, and as such we created a palette of semantic paint from which the user can choose a particular label. Given the limited nature of this label select system, we found it was sufficient to make the range of L and C equivalent, thus reducing λ to a binary indicator variable which is 1 when $L = C$ and 0 otherwise. In text-based systems, a more complex λ would be appropriate. The ρ distribution is provided by the ILP, created using the image categorization introduced in Chapter 3 and refined in Chapter 4, and is pre-computed and stored in a database along with γ for every database image. It should be noted that due to our simplification of the λ variable, Equation 7.1 collapses to

$$P(d) = \prod_{k \in K} (\rho[d, w_k]\gamma[d, k, w_k]) \prod_{u \in U} \left(\sum_{w} \rho[d, w]\gamma[d, u, w] \right) \tag{7.2}$$

thus allowing ρ and γ to be pre-multiplied. Images are segmented by the segmentation forest system of Chapter 4, and then scaled to a square grid with 32 cells to a side, thus requiring $4096 \times Z$ bytes per image to store (where $Z = |\{c\}|$ is the number of cell categories), which in the case of the MSRC21 dataset is 90KB (due to 21 categories + 1 void category). This number depends mostly upon the grid resolution. This is a case of a space/time/performance tradeoff, where less storage space will increase speed at the cost of search performance. The computations for the search are independent given the observed values of L_i and thus can be parallelized with considerable ease. In our experiments, a search of a database of 591 images took an average of 63ms to perform using one core of a Pentium Core 2 Duo 2.4GHz computer, using our un-optimized C# implementation.

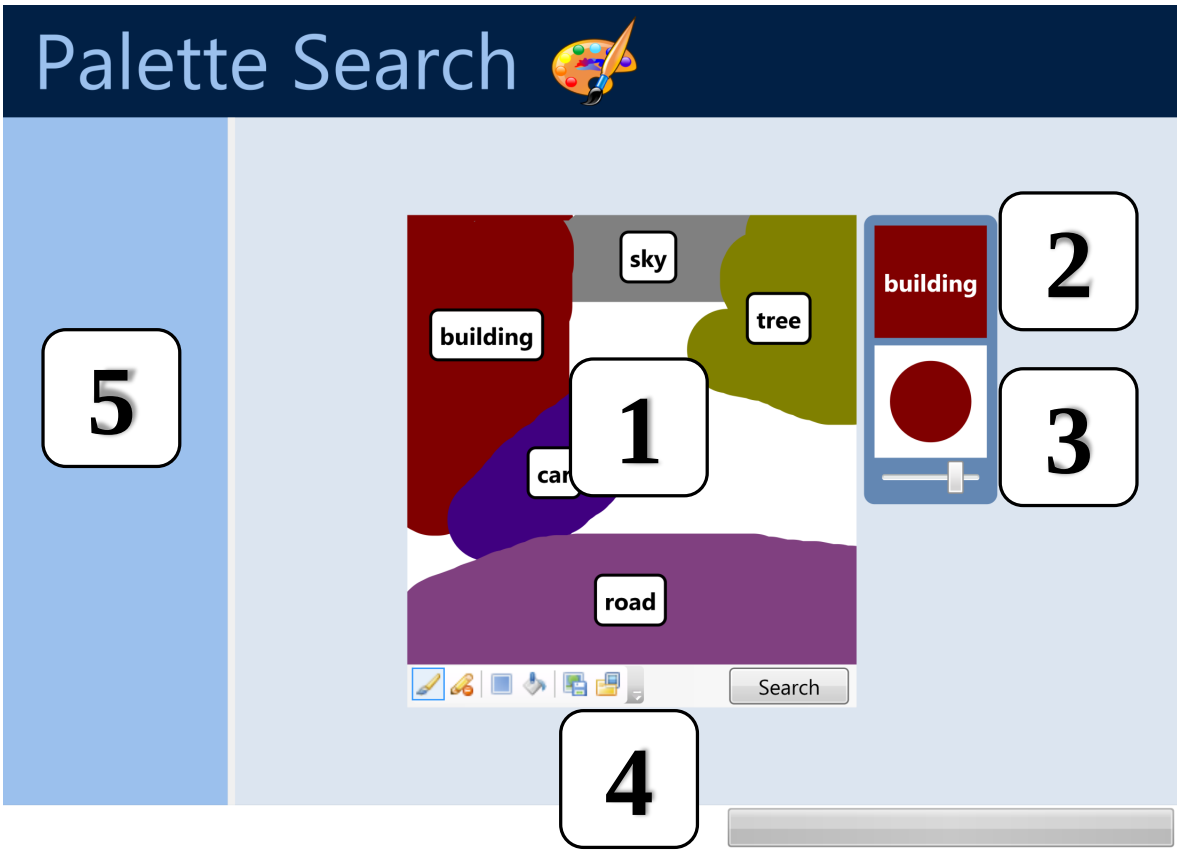


Figure 7.1: *The User Interface*. The user interface is much like a bitmapped paint program, but the paint corresponds to *semantic categories*. The numbers correspond to the items listed in Section 7.3. This is the default mode of the program, during which the user creates a query.

7.3 The User Interface

As can be seen in Figure 7.1, the user interface is clean and uncluttered, presenting a simple paint interface to the user. The various parts of the program are highlighted with numbers in the Figure. The parts of the UI are:

1. Query Canvas
2. Label Palette
3. Brush Size Tool
4. Query Toolbar
5. Results Pane

The *Brush Size Tool* allows the user to easily change the brush size. The *Query Toolbar* contains useful commands for clearing the canvas, filling it with a particular color of semantic paint, the toggle for going between painting and erasing modes, buttons for saving and loading searches and the button which initiates a search. The gray bar in the bottom right corner and the blank space in the bottom left is used to notify the user of progress on tasks running in the background and to give status updates.

7.3.1 Query Canvas

The Query Canvas can be seen in more detail in Figure 7.2. Here the user has painted a query consisting of five things. We chose a painting-program approach based on the fact that it is a well-known and established interface with which people are familiar and that it provides a very explicit way of allowing the user to specify the semantic composition of their target image. However, this aspect can also hurt us, as users sometimes believe that

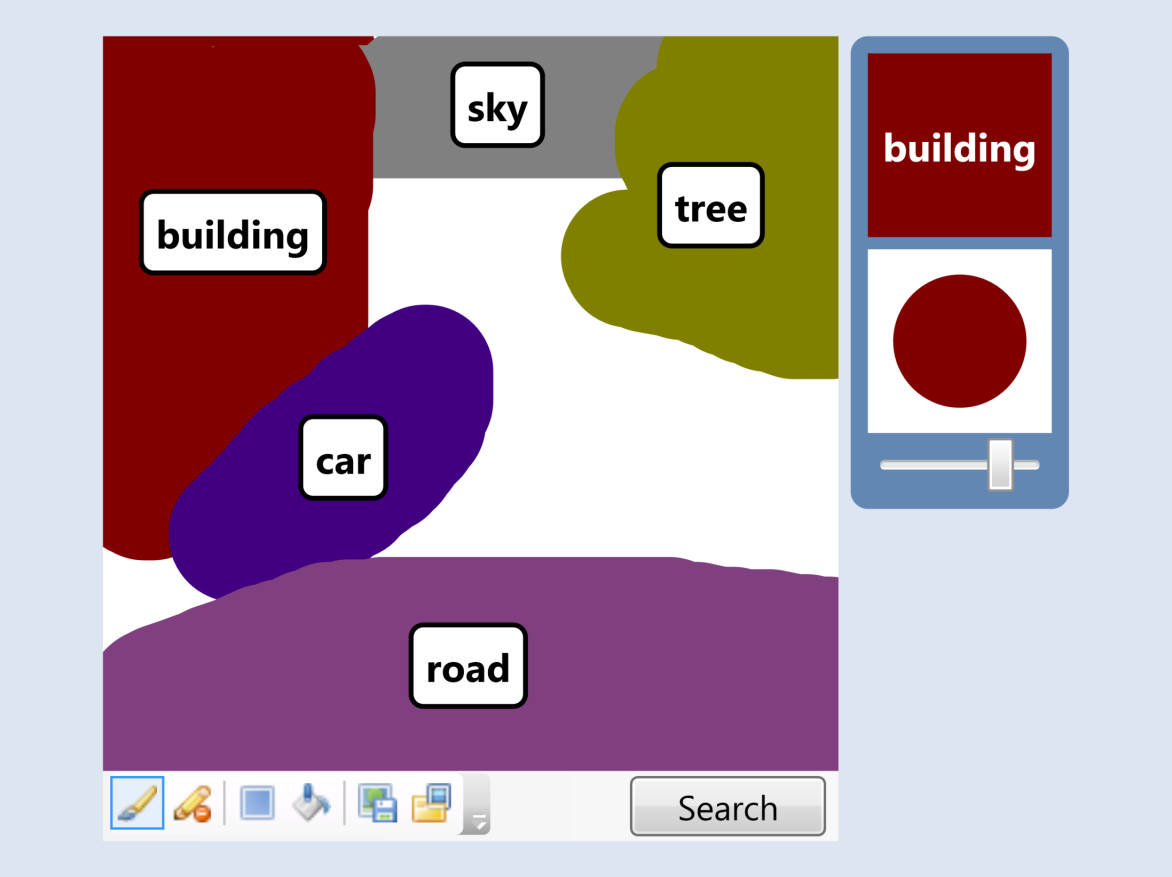


Figure 7.2: *The Query Canvas*. We can see here that the user has put together a complex, multi-object query. He has been very specific about where certain objects are. The labels float with the paint, to give feedback so that the user is aware of what each color corresponds to. Unpainted areas of the canvas are allowed to vary between the provided semantic categories during the search.

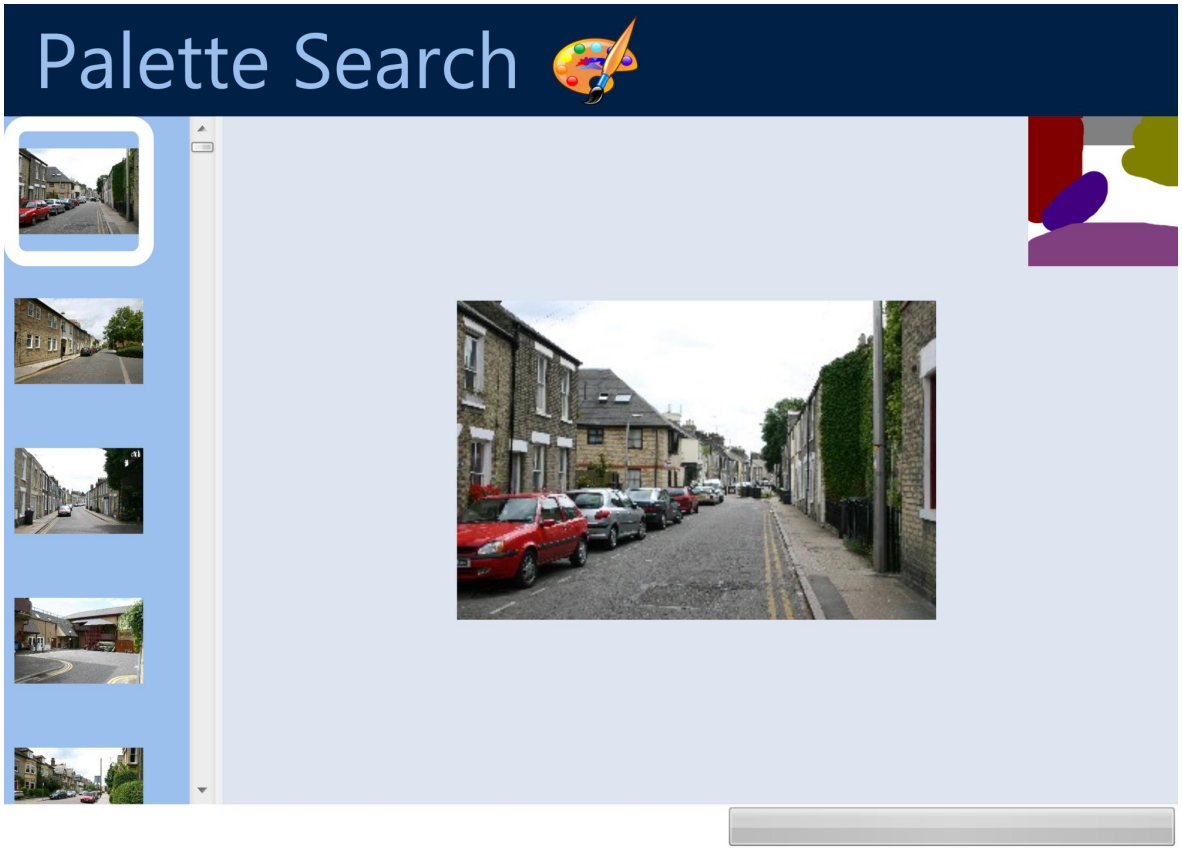


Figure 7.3: *Results View*. The user is shown results in the scrollable pane to the left of the UI. The space allocated to this pane can be adjusted by way of the grip separating it from the main area. As the user selects images from the pane they are shown at a higher resolution in the main area. A thumbnail of the query at the top right gives the user a reminder of what they requested and enables them to match the result images to what they provided. By clicking on this thumbnail the results pane clears and the query canvas returns for modification of the existing query or the creation of a new one.

they must sketch out a drawing of the scene using all of the colors to make a representation of each object (*e.g.* using brown for the trunk of a tree and green for its leaves, instead of the semantic paint 'tree' over the region). The addition of labels on the paint and a brief tutorial kept users from making this understandable mistake. The drawing interface is one which is intuitive to use with a mouse, but truly lends itself to those users of devices whose main input mode is a stylus.

7.3.2 Results View

The application in results view is shown in Figure 7.3. The user is presented with a list of results on the left, and by clicking on one of them is shown a larger version in the center. The amount of space allotted to the results browser can be changed by way of the divider between the two, with the results smoothly adjusting to fill the space, and the full-size version shrinking to accommodate if necessary. At the top right is a thumbnail version of the user's search, and by clicking on it they will clear the results and bring back the palette, with their search as they left it. Thus, modifications to the search can be made easily if the user wants to adjust it to get better results.

7.3.3 Label Palette

An example of the Label Palette when it is opened can be seen in Figure 7.4. Note that each color of semantic paint is shown in a grid. The palette is opened by one click and the paint color chosen by one click, an interface ideally suited for users of stylus-input based machines. Due to the number of categories and the need to have different colors for each, it is not possible to have the colors correspond to each category, and thus the categories are arranged alphabetically to make them easier to locate.

7.4 Experiments

The goal of our experiments was to show that semantic composition was an effective means by which to perform QBSC searches into an image database. As such, the images in that database had to be ground truth labeled in order to evaluate performance, which ruled out the majority of the current CBIR datasets. Also, the inability of current methods to perform such a search made it particularly difficult to compare performance between the

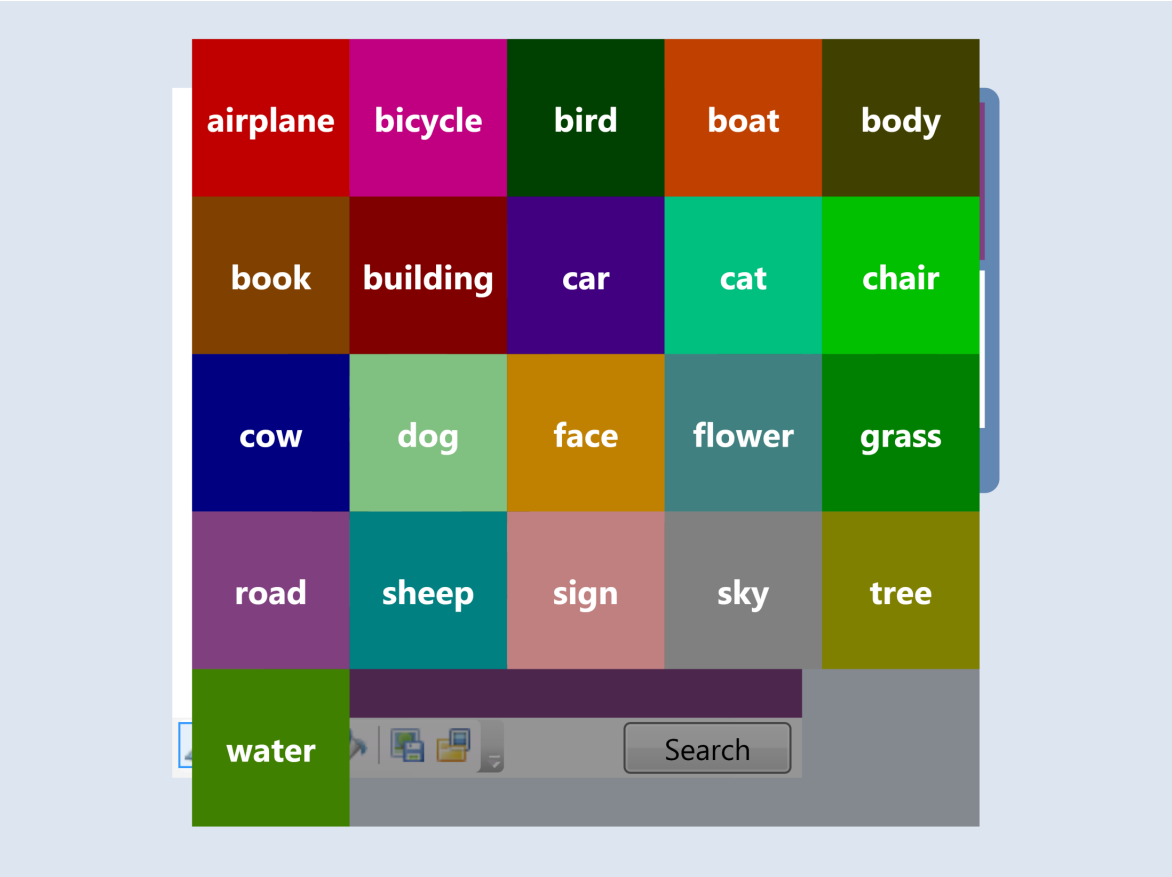


Figure 7.4: *The Label Palette*. The label palette consists of a rectangular grid of colors. The user clicks the palette button once to bring up this grid, and makes his selection with one click on the paint color. The labels are in alphabetical order for convenience. With minimal practice, this becomes a very efficient and fast method of switching input modes.

techniques. Therefore, we performed our quantitative experiments with the semantic composition search which we have described, with a random ordering of the database images to compare against along with a QBSE algorithm, in this case the responses from the ILP. We calculate $P(D)$ as the

$$P(D, \{L_q\}) = \prod_{q=1}^{Q} P(D|L_q) \tag{7.3}$$

$$P(D|L_q) \propto P(L_q|D)P(D) \tag{7.4}$$

where $\{L_q\}$ is the set of user-provided labels, $P(L_q|D)$ is the probability of the label computed by the ILP for an image D and $P(D)$ is a uniform prior over documents in the database. In order to avoid zero probabilities in $P(D|L_q)$, a Dirichlet prior over labels is employed when computing $P(L_q|D)$.

Performance was evaluated using the area under the ROC curve metric and average precision. The AuC value is obtained by integrating under the ROC curve, defined as $\{(\text{fpr}_i, \text{tpr}_i)|\forall i\}$ where i is a position in a ranked list of results. The false positive rate at rank i is defined as:

$$\text{fpr}_i = \frac{\sum_{r=1}^{i} 1 - c(r)}{\sum_{r=1}^{N} 1 - c(r)} \tag{7.5}$$

where N is the size of the list and

$$c(r) = \begin{cases} 1 & \text{if } r \text{ is a correct example} \\ 0 & \text{otherwise} \end{cases} \tag{7.6}$$

Similarly

$$\text{tpr}_i = \frac{\sum_{r=1}^{i} c(r)}{\sum_{r=1}^{N} c(r)} \,. \tag{7.7}$$

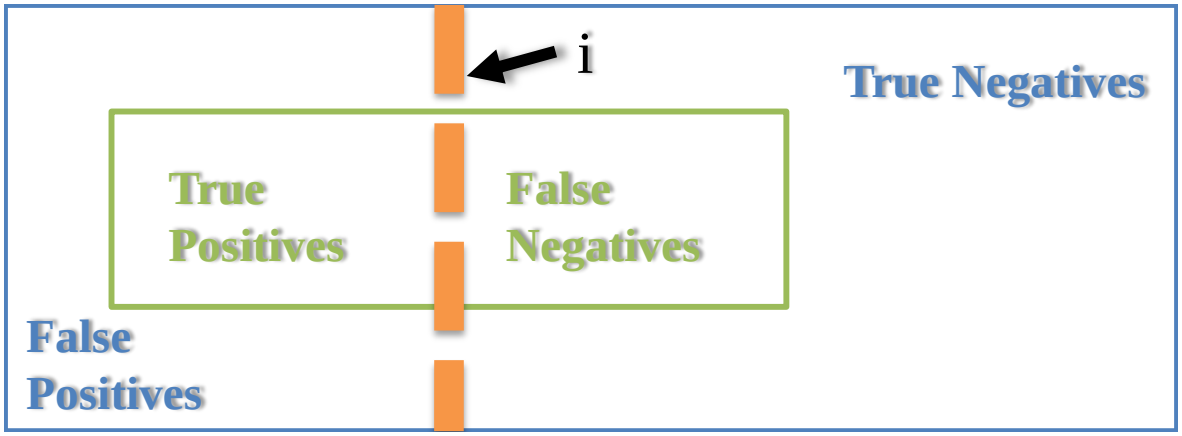


Figure 7.5: *Precision and Recall*. The green square is the set of correct results and the blue square is the entire database. The orange dotted line denotes a particular rank in a ranked list, where everything in the diagram to the left of the line has been returned by a search algorithm. The Precision is the proportion of the area of the green square to the left of orange line to the area of the blue square to the left of the orange line. The Recall is the proportion of the area of the green square to the area of the blue square.

A Precision/Recall curve consists of points $\{(R_i, P_i)|\forall i\}$ where precision is measured as:

$$P_i = \frac{\sum_{r=1}^{i} c(r)}{i} \tag{7.8}$$

and $R_i \equiv \text{tpr}_i$. For further aid in understanding, precision and recall are explained graphically in Figure 7.5. The Average Precision performance metric is a smoothed integration under the PR curve, computed as

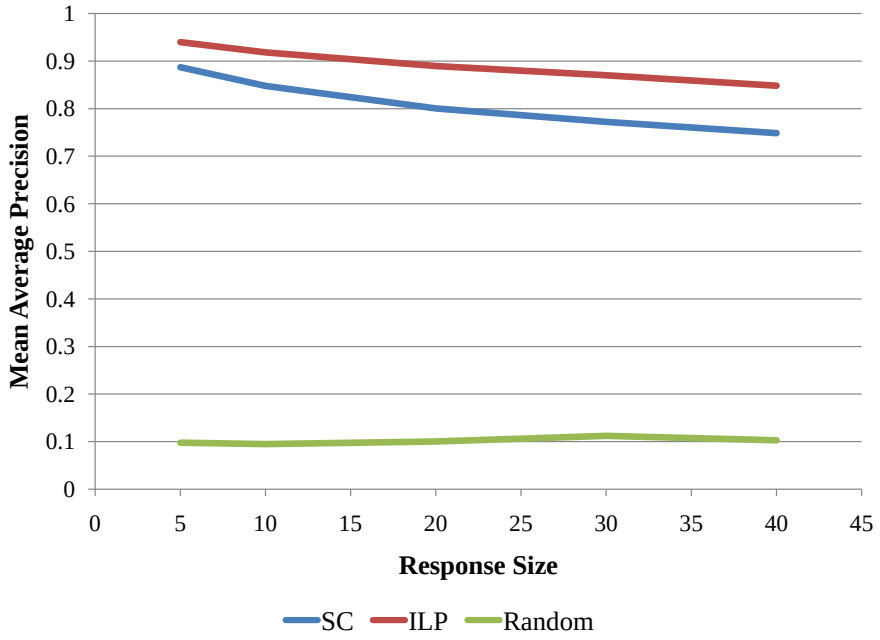
$$AP_i = \frac{1}{\sum_{r=1}^{N} c(r)} \sum_{r=1}^{N} P_r c(r) \tag{7.9}$$

These are the standard measures of performance for Information Retrieval, however as noted by [95], recall in particular and standard IR performance measures in general begin to be less effective determiners of system performance for image retrieval the larger the response size becomes. That concern combined with the comparatively small size of one of the test databases (591 images) influenced us to constrain the experiment to the top 40 responses.

We performed 3 quantitative experiments. All searches were performed using the ground truth labels of the 256 unseen test images as the query. For details on the MSRC21 dataset, see Appendix B. In the first experiment, we compare the three algorithms in a content-only search, where a true positive is defined as an image which shares all of the labels of the query image. Thus, if the query is of a sheep standing in grass, any image with sheep and grass is considered a possible result. In the second, we compare all three algorithms in a compositional search, where a true positive is defined as an image whose semantic labels overlap the query's labels by at least 75% for all categories individually. So, if the query is of a sheep standing in the grass, any image which has sheep pixels at least 75% the same as the query image and grass pixels at least 75% the same as the query image is considered correct. Naturally, this is a much harder task, and one which requires a pixel-level understanding of the semantic content of an image. In each experiment we search the database with the ground truth segmentation of each test image used as its semantic composition query. We then record the mean AP and AuC over all searches for response sizes of 5, 10, 20, 30, and 40. Our third experiment consisted of varying the percentage of pixels that match required to be considered a true match, to see how the two algorithms degrade as the performance requirement increases.

The results for the content-only experiment are shown in Figure 7.6. As is to be expected, the QBSE algorithm performs best, closely followed by the semantic composition. This is for several reasons. First is the assumption in the model that every known pixel is of equal importance. One potential drawback of this system is that the importance of a category to the model is directly related to how many pixels the user provides a known label for. Thus, if the user provides 1000 grass labels and 24 sheep labels, the system will (arguably correctly) concentrate on images which match the grass moreso than those which match the sheep. Since the positive results for this search contain images which may have the

(a)



(b)

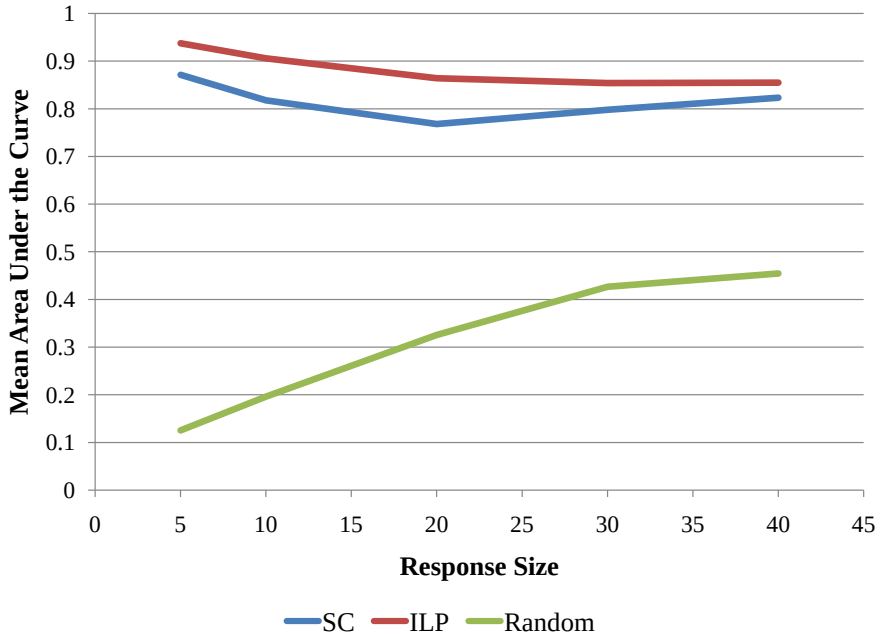


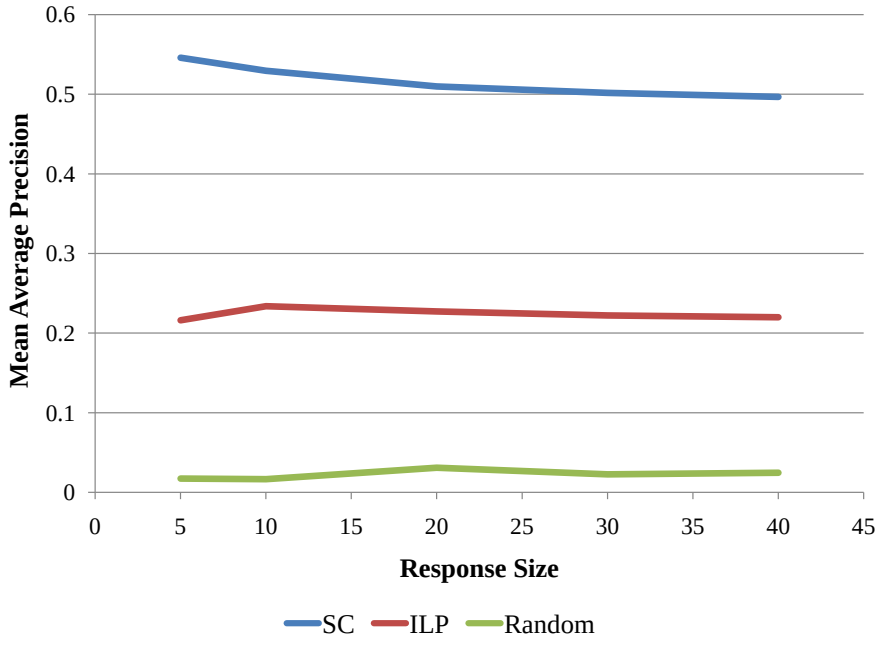
Figure 7.6: *Content-Only Search Results*. As is to be expected, our Query by Semantic Example algorithm (denoted ILP here) slightly outperforms semantic composition (denoted SC). Random performance is included for reference. The reasons for this result are presented in the text, but in brief semantic composition is bound by design to give great emphasis not just to semantics but to location. Thus, as far as it is concerned images which have the right semantics but the wrong composition are incorrect, though by the metric for this task they are correct. This is not the case for the QBSE algorithm, which only acts at the global image level. Performance for average precision (a) and area under the ROC curve (b) is shown.

same categories but in a completely different configuration than the one specified, it is very natural that the semantic composition system will perform slightly less due to ranking those images lower than images which are partial matches but have a subset of object classes in the right positions. An example of this can be seen in Figure 7.10 with the dog images, where a search with two or more items that are in an uncommon positional combination (a dog and a boat) results in the top images returned being strong partial matches for one of the objects. These images, though near matches in terms of position, are not deemed true positives for the search due to the constraint that all classes must be present.

The graphs in Figure 7.7 show the performance of the search algorithms on the compositional search task. Here we see the clear improvement gained by using the semantic composition on this much harder task, as the content-only method fails to retrieve images which have the right content in the right position. Retrieval results for successful and unsuccessful queries on the MSRC21 dataset are shown in Figures 7.9 and 7.10. The success images shown are the top four results as ranked by average precision (with $AP = 1$) and the failure images are the bottom four (with $AP \approx 0$). As can be seen, the "75% of pixels the same" correct metric is quite punishing, but semantic composition is still able to perform very well. In Figure 7.8 we show the effect of changing the metric for correct images from at least one pixel the same to 90%, which shows the point at which just returning images with the same semantic content does not satisfy the user's query as well as semantic composition is about 12% pixels the same.

The system has three main failure modes, shown in Figure 7.10. (a) is a typical example in which a semantic category with a strong signal dominates the unlabeled pixels, as well as demonstrating the inherent scale issues in the system. While scale can be constrained to a certain amount, with the face category in particular it can result in rankings like these. (b) and (c) show the main failure mode, in which two or more semantic categories which occur

(a)



(b)

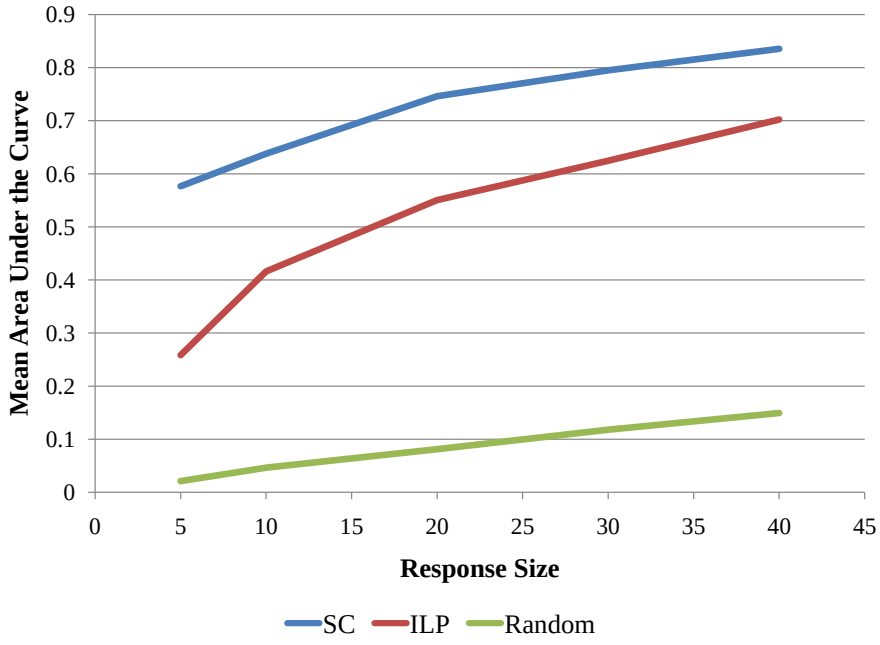


Figure 7.7: *Compositional Search Results*. Here we see the remarkable improvement of Query By Semantic Composition (SC) over Query by Semantic Example (ILP) on this very difficult task. Correct images were those which shared 75% of pixels with the query for all semantic categories. The average precision (a) in particular shows the steady performance of the algorithm, even as the response size becomes very small. Performance for average precision (a) and area under the ROC curve (b) is shown.

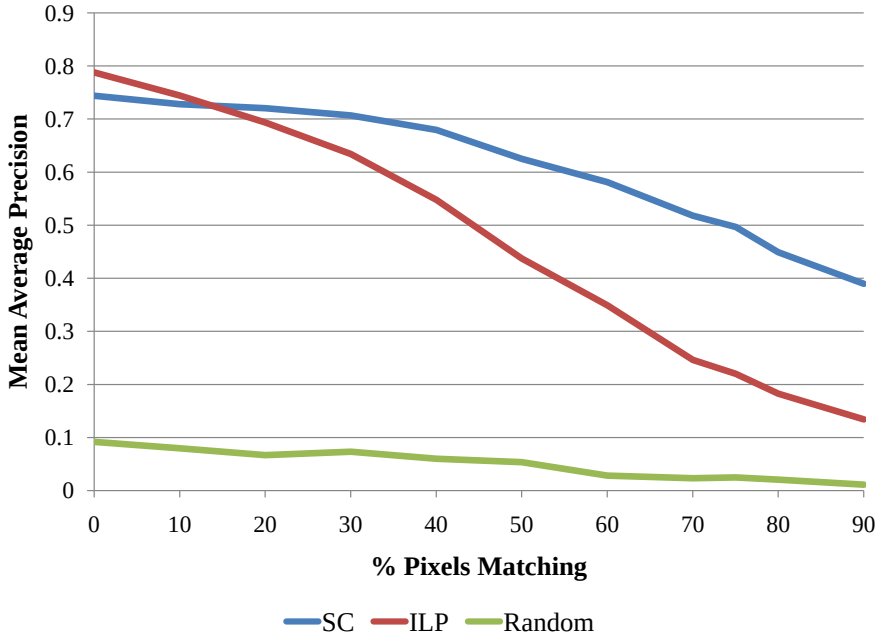


Figure 7.8: *Increasing the Required % of Pixel Match*. We see here that very early on Query by Semantic Example (ILP) is unable to satisfy the requirements of the query. This is to be expected to a certain extent, given that it is position agnostic, but demonstrates what our Query by Semantic Composition (SC) algorithm is able to achieve in this dataset over current methods which do not take into account the positional semantics of images.

together only rarely in a certain composition are specified in the query, and thus the results concentrate on strong matches for a subset of the categories. (d) shows erroneous matches as a result of faulty information from the segmentation algorithm (*e.g.* the man in the jacket is not a sign).

In addition to the MSRC21 results we provide results on the Scenes dataset. Due to over-labeling on the part of anonymous users and inaccuracies, the quantitative measures score very low. They show the same trends, however, as can be seen in Figure 7.11. Results of the compositional search can be seen in Figures 7.12 and 7.13, following the same methodology of giving the four highest and lowest searches. The failure cases here echo what we saw in the MSRC21 dataset. All four show situations in which large regions of one label have dominated the matching process to the exclusion of other labels. In (a) and (c) the segmen-

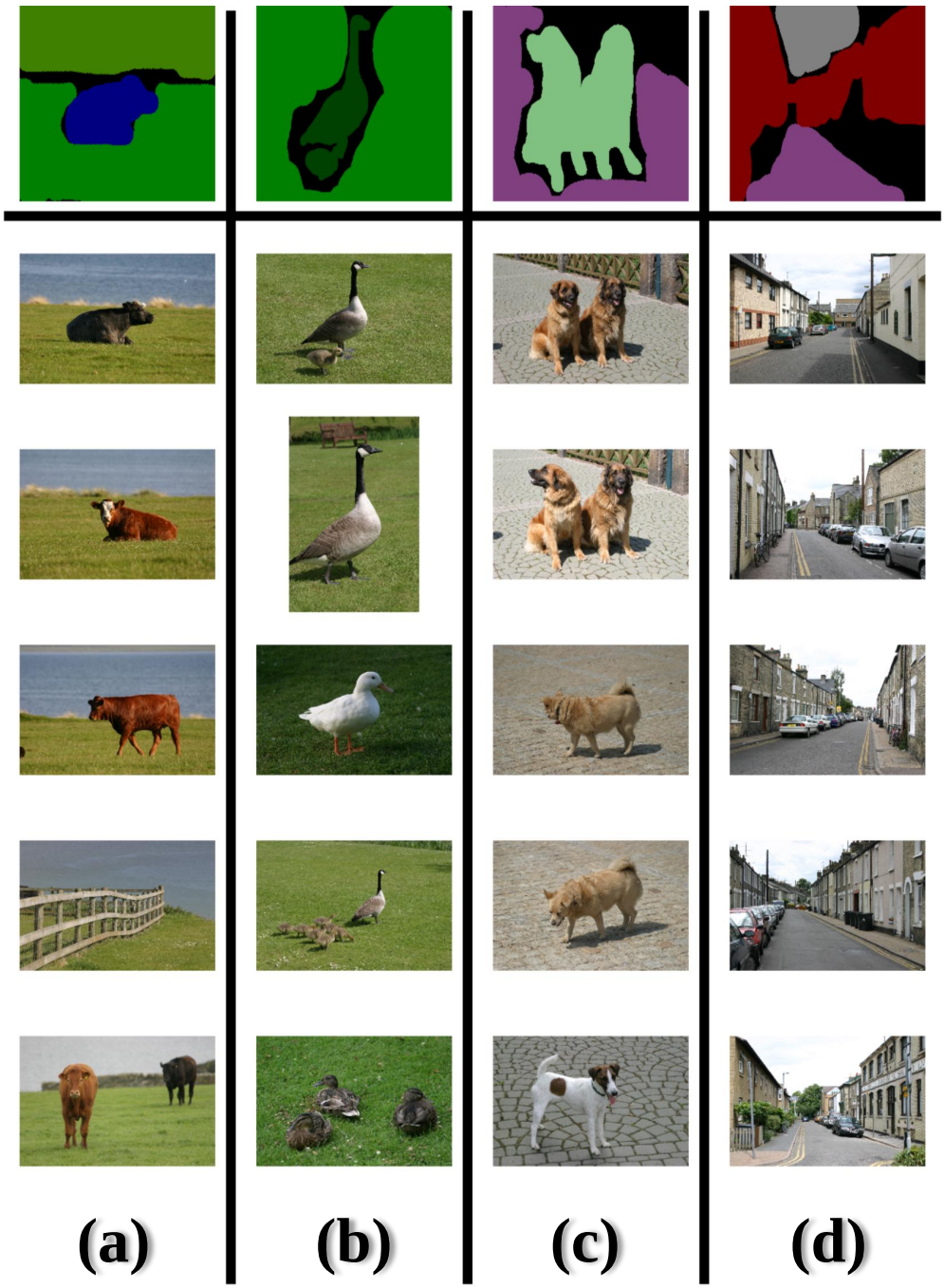


Figure 7.9: *MSRC21 Example Compositional Search: Success*. Shown here are four searches, each of which had an average precision and AuC of 1 over the entire dataset with 75% of pixels matched required for correct results. In total, 49 of the 256 searches performed had an AP of 1, with 130 having APs above .5 over the entire dataset. (a) shows a search in which there were only three correct results, and as expected the next two results contain subsets of the objects in the query in the right locations.

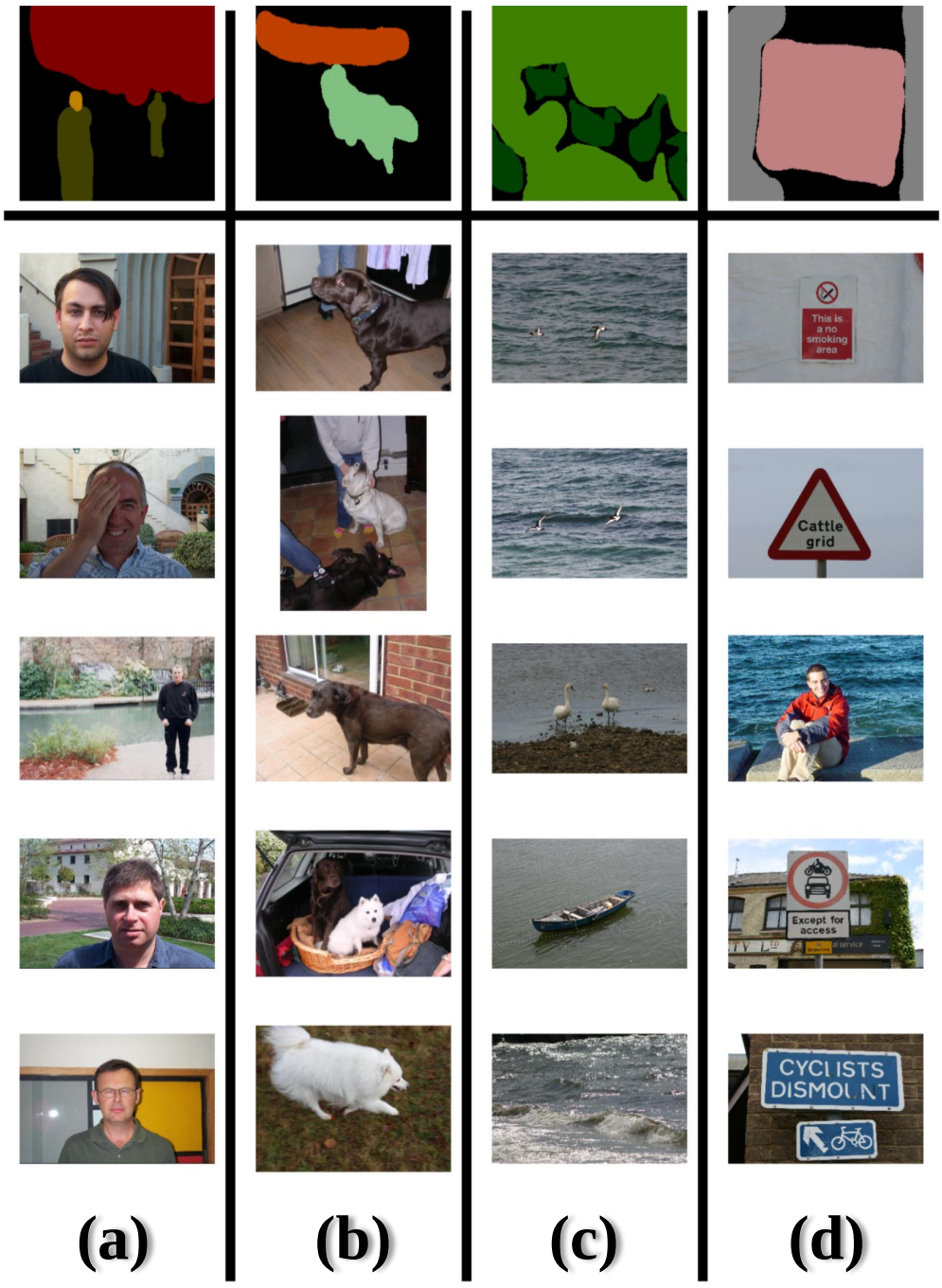


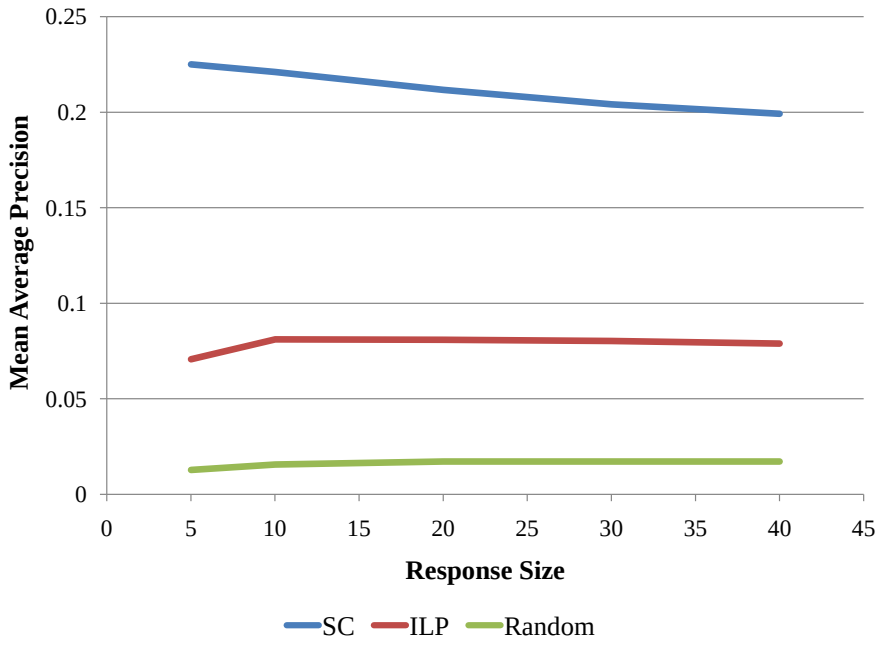
Figure 7.10: *MSRC21 Example Compositional Search: Failure*. Shown here are several standard failure cases of the system. These four searches are drawn from the bottom 10 searches, all of which had an AP of approximately zero. For a full discussion of the failure cases, please refer to the text.

tation algorithm has incorrectly labeled parts of the image, resulting in incorrectly ranked results. A typical problem with this dataset can be seen in (c), where a very small area of the image is marked as "sign", the result of zealous anonymous labelers. This kind of labeling actually hurts training, and drives the accuracy numbers down during evaluation. This is an area where a hand-crafted dataset such as the MSRC21 dataset is very forgiving, by providing buffer zones labeled 'void' at the boundaries between different categories and by ignoring obscured or small objects in the ground truth data. At the same time, these results demonstrate the ability of the system to still perform well even on large, varied datasets.

7.5 Summary

The Palette Search system is an exciting new way of searching images, enabling the user to specify both what is in their desired image and where it should be. It is a kind of search only made possible by the presence of an accurate semantic segmentation algorithm, such as we have presented in Chapter 4, and shows one of the possible new directions for future research in content-based image retrieval. In particular, it provides a powerful interface for image search which eschews text input in favor of a graphical, point-and-click approach particularly appropriate for image search. In the next chapter, we will discuss our findings and possible future directions for this research.

(a)



(b)

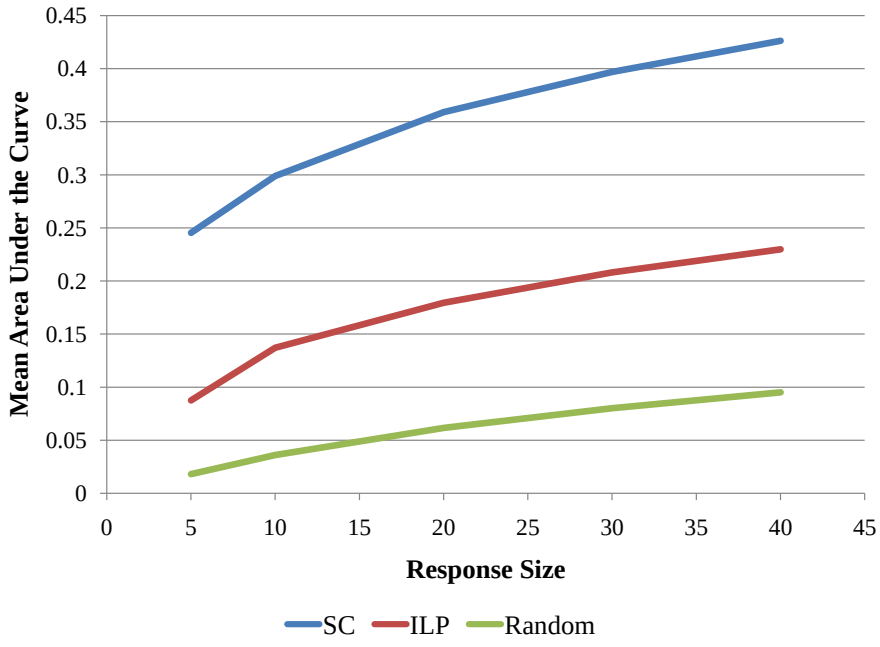


Figure 7.11: *Scenes Compositional Search Results*. Here we see the remarkable improvement of Query By Semantic Composition (SC) over Query by Semantic Example (ILP) on this very difficult task. Correct images were those which shared 75% of pixels with the query for all semantic categories. The average precision (a) in particular shows the steady performance of the algorithm, even as the response size goes to very small sizes. Performance for average precision (a) and area under the ROC curve (b) is shown.

Figure 7.12: *Scenes Example Compositional Search: Success*. Shown here are four searches chosen from the top 100 as ranked by AuC, with 75% of pixels matched required for correct results. In total, 140 of the 2202 had an AP of 1, with 1475 having an AuC of .9 or above over the entire dataset.

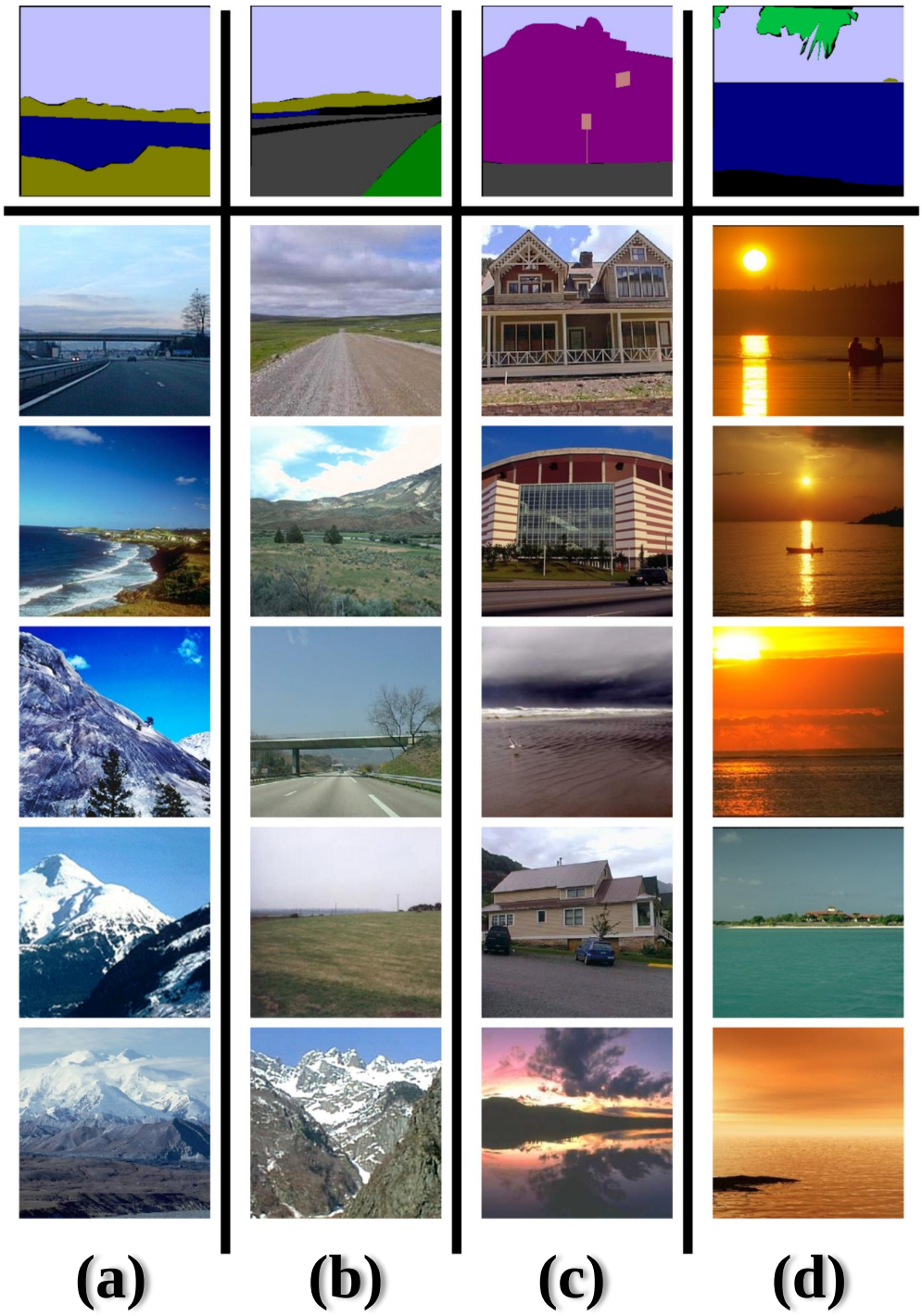


Figure 7.13: *Scenes Example Compositional Search: Failure*. Shown here are several standard failure cases of the system. These four searches are drawn from the bottom 100 searches as ranked by AuC. For a discussion of failure cases, please refer to the text.

CHAPTER 8

CONCLUSION

Throughout this thesis we have discussed various new solutions to standing problems in computer vision for which we have furthered the state of the art, culminating in a new form of image search. We will now discuss our findings, the limitations of our techniques, and directions for future research.

8.1 Findings

The first part of the thesis concentrated on our novel solution to the problem of *semantic segmentation*. In Chapter 2 we examined the state of the art in image categorization through the use of bags-of-words models, this being the dominant manner in which semantics is incorporated into image search. We found that though bags-of-words models throw away spatial information, they remain quite effective, especially when independent cues are combined together. We then introduced a novel visual inference framework, the *semantic texton forest*, in Chapter 3, and demonstrated its uses for image categorization. The semantic texton forest is tailored to the task of semantic understanding of visual data, and is able to learn from the data itself what features are most useful for pixel categorization and labeling. We explored the effects of the various training parameters on pixel categorization accuracy, and showed how it was possible to train a forest for pixel classification in the absence of ground-truth

labeled data. In Chapter 4, we demonstrated the power of semantic texton forests for semantic segmentation. We showed that we have obtained the state-of-the-art for performance on this task as compared to the techniques of Shotton *et al*. [92] and Verbeek *et al*. [103], while developing a technique which is incredibly fast and efficient due to its tree-based structure for feature extraction and label inference. In Chapter 5, we explored the theoretical model of the semantic texton forest and demonstrated that it can be viewed as a kernel density estimator, and as such can be used as part of a Bayesian pixel categorization model. We then used that model to infer an image's annotation, and demonstrated that it is possible to significantly improve pixel categorization accuracy by performing further inference on the model.

In the second part of the thesis, we concentrated on the problem of *semantic image search*, more specifically the ability to perform *Query by Semantic Composition*, which we present as a challenging new research direction for content-based image retrieval. We introduced a new method of query representation, the *semantic composition*, in Chapter 6. We described the probabilistic foundation for image search and comparison which it provides, and placed it in the context of the continuing development of content based image retrieval. This culminated in Chapter 7, where we introduced the Palette Search system. Palette Search enables the user to specify the semantic content of their target image and the composition of that content, *i.e.* where in the image plane it is located. Using the semantic segmentation technique developed in Chapter 4, we automatically segmented a database of images and performed queries into this database using the semantic composition model. We showed that our technique significantly improves over a Query by Semantic Example technique, which represents an approximation of the current state of the art given a lack of existing results by standard CBIR algorithms on our datasets. While there exist techniques which enable the user to specify colors, textures or primitive shapes for regions of an image, ours is the first

system which enables a user to specify multiple semantic regions, and the first to allow the user to do so using an intuitive paint-like interface.

8.2 Limitations

The techniques we have presented are a step forward in our ability to understand the underlying meaning of visual content, but there is still much work to be done in this area. One of the major limitations of the technique presented in Chapter 4 is its reliance on ground-truth pixel labels. This data is expensive to obtain and therefore scarce. While we are able to exploit what data is available very effectively, and have developed means by which to create ground truth labeled data from anonymously labeled data on the Web (see Appendix B for details), this remains a serious limitation of the technique. Indeed, the second-level forest which we employ cannot be trained without it. While the model we presented in Chapter 5 can potentially be used to overcome this limitation by unfixing $P(C|E)$ and fitting the distributions by way of observed image topic labels X, this still requires trusted image topic labels, which while more plentiful are still limited in their scope. The next generation of semantic segmentation algorithms must be able to be trained in a partially supervised, if not entirely un-supervised manner.

Query by Semantic Composition, while a novel concept, presents anew old problems in terms of algorithm evaluation. Image retrieval has always been plagued by the question of how to evaluate the performance of algorithms [95], and the inclusion of positional semantic constraints on the problem does not make this question any easier to answer. The gold standard is really evaluation by human users on large datasets, and there has been significant research on how best to perform this [51]. We avoided the problem by using databases with semantic data for all pixels and thus were able to evaluate precisely which images were correct for the user's query, but the true test of ours or any system is its ability to satisfy users

on large, varied datasets, for which ground truth data is unavailable.

8.3 Future Work

Semantic texton forests show incredible promise as a new framework for semantic segmentation. As such, there are many directions which we are interested in exploring:

Real-Time Semantic Segmentation of Video The structure of the STF lends itself well to optimization, in particular hardware optimization using programmable graphics chips. An interesting research direction would then be the semantic segmentation of video in real-time, for use in car/robot navigation or video frame search. In particular, the additional dimension of time allows the introduction of new feature tests to the forest which act on the space-time cube, allowing previous frames to aid in the segmentation of future frames.

Volumetric Semantic Segmentation Following on in the same vein, the use of STFs for volumetric semantic segmentation, specifically in medical imaging, is an area of considerable interest for future research. The ability of the forest to learn the most useful features from the data, the smoothing effect of multiple trees, and the ability to perform tests in three dimensions and across multiple image channels are particularly suited to the very difficult problem of volumetric segmentation of medical images.

Conditional Random Fields While the two-tiered segmentation technique introduced in Chapter 4 incorporates compositional cues into the segmentation process and thus mitigates the effect somewhat, our current model for semantic segmentation makes the very naïve assumption of grid cell independence conditioned on the topic. In reality, adjoining grid cells are connected to each other, and a more accurate model would have dependencies between these cell labels, creating a conditional random field [59].

This would result in cleaner segmentations, enforcing local uniformity in the labels.

Partially Supervised Training We have begun preliminary research on the training of semantic texton forests in the absence of ground truth labeled pixels, but there is much remaining to be done before we are able to approach the same quality of segmentation as in the fully supervised case. Russel *et al*. have begun an interesting line of research in [84] which uses multiple photometric segmentations to determine object boundaries. This method could easily be adapted for use with STFs, and is a promising direction for this research.

Having proposed Query by Semantic Composition, we are interested in seeing the development of new datasets and algorithms which will further its development as a new form of image search. Our efforts at the automatic conversion of image label data from LabelMe [85] are a step in the right direction, and the existence of these datasets should be exploited for use in the CBIR community to a greater extent. In particular, we would be interested in launching a large-scale image search user study to determine preferences for different kinds of image search in general, and QBSC in particular.

8.4 Final Remarks

Visual object recognition continues to advance dramatically, with our work being but one of a series of exciting strides forward in the past few years. The addition of semantic segmentation as a new category in the Pascal VOC2007 object challenge [28], is an indication of how the community is attempting more and more difficult tasks as a whole. Indeed, the remarkable performance of algorithms in the image classification task of this most recent Pascal challenge on thousands of difficult images from 20 categories is a sign of how far we have come. To put this in perspective, however, the human visual system can recognize tens of

thousands of different object categories in a fraction of a second, with astounding accuracy. Our field still has a very long way to go before we can approach that order of magnitude for performance.

Similarly, while CBIR algorithms continue to improve, there remains a barrier between the community and actual image search algorithms in common use. This barrier is partially due to inertia of current search paradigms, but also due to a lack of query innovation. As long as text input remains the main form of image query, caption-based methods will continue to dominate the field. Query by Semantic Composition provides a new paradigm for image search that is uniquely achievable using image understanding. As the content of the web becomes increasingly separated from its representation and display, active and involved search paradigms and dynamic content creation will demand intelligent agents which understand all forms of media. This provides a unique opportunity to create demand for better image search capabilities, and with the explosion in the quantity of digital image and video data we finally have the raw material available to create these agents. Only by developing generative image models which can be trained in an unsupervised manner can we exploit this data fully. We believe that semantic texton forests are an important step towards these learning techniques, and that semantic composition provides a model for image search and retrieval which can incorporate them as they become available.

APPENDIX A

REPRESENTING VISUAL DATA

Visual data is rich with varied kinds of information. The raw pixels of an image tell us little on their own besides distributions of color, light and shadow. They themselves are merely a discrete representation of a continuous visual signal, a two dimensional representation of a three dimensional scene. Thus, it is necessary to process the pixels in order to gather more information about an image, and to store it effectively and efficiently for further tasks. In this appendix, we will discuss the nature of digital color imaging and its limitations, and then perform a review of interest points and descriptors and how they can be used to sift through visual data for nuggets of useful information which can inform the higher level tasks discussed in the thesis.

A.1 Color Spaces

Color is a complex property, and storing it digitally is difficult to do accurately due to the fact that different imaging devices will produce different values for the same color, thus making device-independent storage only possible through calibration. Even if the values are being encoded accurately, however, the wavelength of light emitted from a surface changes based on a variety of environmental conditions such as the viewing angle or the color and strength of the light source(s), though the properties of the surface do not. Thus, it is possible

to get two different color readings for the same point on a surface from one moment to the next. In this section we will review different methods of storing color in common use and how effective they are at dealing with this problem in relation to their ease of use and understandability.

A.1.1 RGB

The *RGB*, or *R*ed *G*reen *B*lue, color space is the simplest method of color data storage and by far the most widespread in computational applications, and has its root in the first efforts with color photography by James Clerk Maxwell in 1861 [43]. The standard method uses 8 bits per color resulting in a 24-bit representation with 0 indicating no amount of a particular color and 255 indicating the maximum amount of that color. For example, (0, 0, 0) is black and (255, 255, 255) is white, with red being (255, 0, 0). The intuition for *RGB* is based somewhat on the human vision system, in which three cones respond to yellow-green, green and blue light to varying degrees, mixing to create the various colors we perceive [47].

Such a system is said to use additive color, since three or more primary colors are combined to create a gamut of colors. It should be noted that *RGB* does not dictate the nature of the red, green and blue that are to be mixed, thus making it possible for the same *RGB* value to result in two completely different colors on two different devices, a significant problem if we are trying to learn color statistics for the purpose of computer vision. For a depiction of the RGB color space, refer to Figure A.1, a cube in which each of the three dimensions represents values for red, green or blue. As can be seen somewhat in this depiction, the values for a particular color, such as "green", are spread over a large area of the cube as intensity varies. A more useful color space would be one which separates color from intensity.

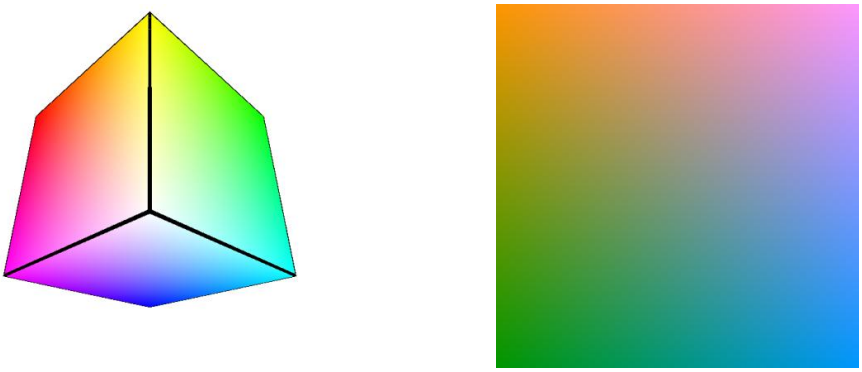
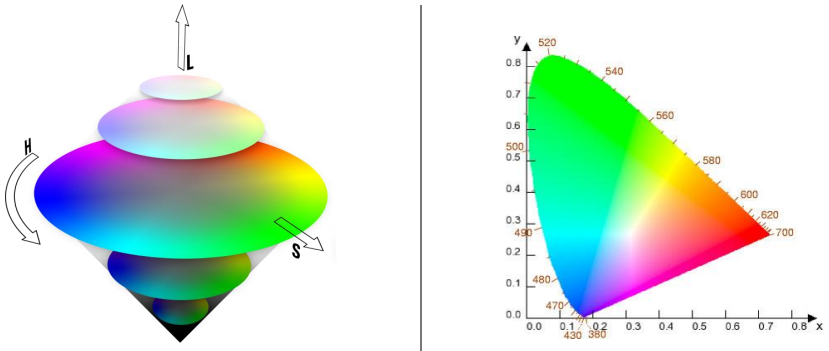


Figure A.1: *RGB Color Cube and Slice* On the left is a rendering of the *RGB* Color Cube as shown from the perspective of the white corner (black lines added for clarification of edges). On the opposite side of the cube is the same view but with each of the colors darkening to black at the point. The slice is taken at Green = 150, with Red increasing horizontally to the right and blue increasing vertically upwards. Notice how any particular color is spread across large areas of the cube, both on the surface and internally.



(a) (b)

Figure A.2: *(a) HSL Double Cone* The *HSL* double cone. This structure captures two main human intuitions about color. First, as Lightness increases to its maximum or decreases to zero, color (Hue) and Saturation become decreasingly important and harder to distinguish. Second, color is separated and forms a color wheel in which similar colors are close together.

Figure A.3: *(b) Lab Gamut* Though this representation is sub-optimal (the actual color space cannot be represented using printed colors alone, what is shown is an approximation) this shows the range of colors which can be represented using the *Lab* color space, shown here as the hues at maximum luminance.

A.1.2 HSL

HSL, or *H*ue *S*aturation *L*ightness, is perhaps the most intuitive of the color spaces, and was introduced by Smith in [96]. Each of the three elements corresponds to one of three aspects of a color: which color it is, how strongly it is that color, and how bright it is respectively. It is designed this way to mirror human perceptions of color, based upon very early work in psychology by Yerkes [108]. If *RGB* is a cube, then *HSL* is a double-cone as shown in A.2, with Lightness on the vertical axis, and Hue and Saturation being angle and radius on the circular cross-section. For computer vision *HSL* can be very useful, as it separates out intensity from color, making it possible to store color as a distinct entity and analyze it independent of light source intensity. However, it suffers from the same problem as *RGB* in that the perceived distances between colors are not properly reflected by the geometric properties of the space [50].

A.1.3 CIELab

The Lab color space was first proposed by Hunter [49; 48] and then updated by the Commission Internationale d'Eclairage (International Commission on Illumination) to the L* a* b*, or CIELab color space, depicted in Figure A.3. It was developed as an attempt to create a complete, device-independent color space which mimics the perceived distances between colors determined by the human brain. It is device-independent based on a known white-point (the set color of white under a known illuminant) and is able to encode all colors visible by the human eye (and some which are outside the range as well.) The *L* represents *L*uminance, *a* denotes where on a scale from red to green the color lies, and *b* similarly with blue to yellow. *Lab* is by far the least intuitive of the color spaces, though its completeness, device-independence and Euclidean properties in respect to human perceived color distance

make it very useful for vision tasks [50].

A.2 Interest Points

While each pixel in an image is potentially interesting, in practice there are several specific points in the image which are useful in computer vision tasks. These can be divided into three groups: corners, edges and blobs. The advantages that these points have is that they can be detected reliably in different images of the same scene because they are created by objects within that scene. The difficulty comes in finding the correspondence between two images of the same scene, which is why interest points are often coupled with local descriptors, which describe the area around them.

The three main kinds of interest points are edges, corners, and blobs. Edges mark the boundaries between different areas in the image, for example areas of different brightness levels, or texture statistics. Corners are found at the peaks in the autocorrelation function (often corresponding to geometric corners, or points where several regions intersect, but not always), and blobs are found in the stable centers of uniform regions. Of the three, edges are easiest to extract, corners are the most numerous and blobs the most stable. All three have biological plausibility, *i.e.* there is evidence that there are structures within natural vision systems that perform all three tasks [66].

A.2.1 Edges

Edges are located where intensity values in the two-dimensional image function undergo a sharp change from one state to another, such as from a white square to a black background. These points are the local maxima of the gradient of the image. Canny edge detection [15] is a efficient process that produces a binary edge image (in which every point is labeled as an edge or otherwise) or an edge list from an input intensity image. An example of edge

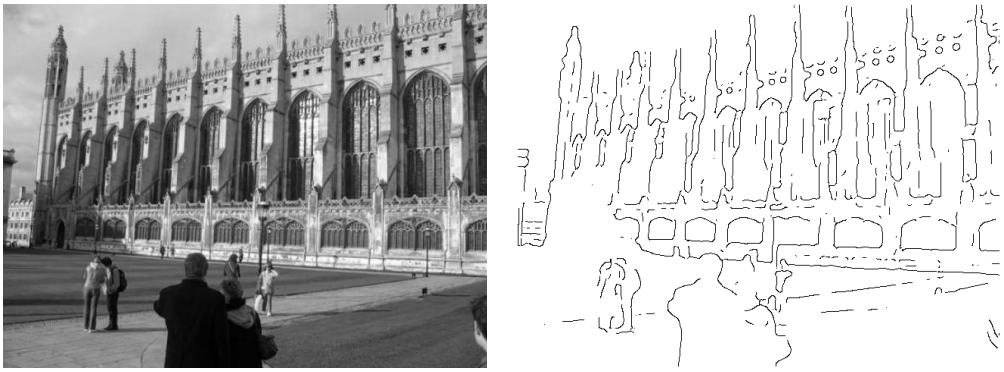


Figure A.4: *Canny Edge Detection* Shown here is an image and its Canny edge map at $\sigma = 3, \tau_1 = .3, \tau_2 = .7$ where τ_1 and τ_2 are the thresholds used for hysteresis. Notice the clean, connected edges and the complete lack of noisy, unnecessary edges in the image.

detection performed on an image is shown in Figure A.4.

A.2.2 Corners

While commonly referred to as "corners", they are in fact maxima in the autocorrelation function [73] and as such often correspond to geometric corners in the scene, but not always. In [39], Harris and Stephens describe a method based on the Moravec corner detector [73] which finds these maxima.

Let $I(x, y)$ denote the smoothed image brightness function. Of interest for corner detection is the matrix

$$H = \nabla I \nabla I^T = \begin{bmatrix} (\frac{\partial I}{\partial x})^2 & (\frac{\partial I}{\partial x})(\frac{\partial I}{\partial y}) \\ (\frac{\partial I}{\partial x})(\frac{\partial I}{\partial y}) & (\frac{\partial I}{\partial y})^2 \end{bmatrix} \tag{A.1}$$

or more specifically, its smoothed version

$$\hat{H} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \tag{A.2}$$

where $\nabla \langle I \rangle = G * \nabla I$, G is a two-dimensional Gaussian smoothing kernel and $\nabla I = [I_x, I_y] = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial x}]$. This matrix describes the gradient characteristics in the window of the

Figure A.5: *Harris Corners* This image displays Harris corner detection, with the 200 maximal corners from the image superimposed in red. Notice how the detector fires not only at geometric corners but also at areas of active texture (*e.g.* the crinkled bag, the tangled wires.)

smoothing kernel G. Its eigenvectors ϕ_0 and ϕ_1 describe the main directions of the gradient with the strength of those directions being directly related to their corresponding eigenvalues λ_0 and λ_1. If both λ_0 and λ_1 are large, this indicates that a patch moved in any direction around the point will have a low correlation to the surrounding pixels and thus the point is labeled as a corner, if only λ_0 dominates the area is an edge, and if both are negligible the area is uniform. As such, this method can be used for both corner and edge detection. The σ used for G determines the scale of the features detected by dictating the size of the neighborhood inspected.

This basic method, though improved by Shi and Tomasi [90] and Noble [74], is still used in many circumstances. One of its limitations as an interest point detector is that it is unable to give an intrinsic scale to the point, that is the scale at which the corner is most strong. This is addressed in the line of work on corner detection by Mikolajczyk [69] by the introduction

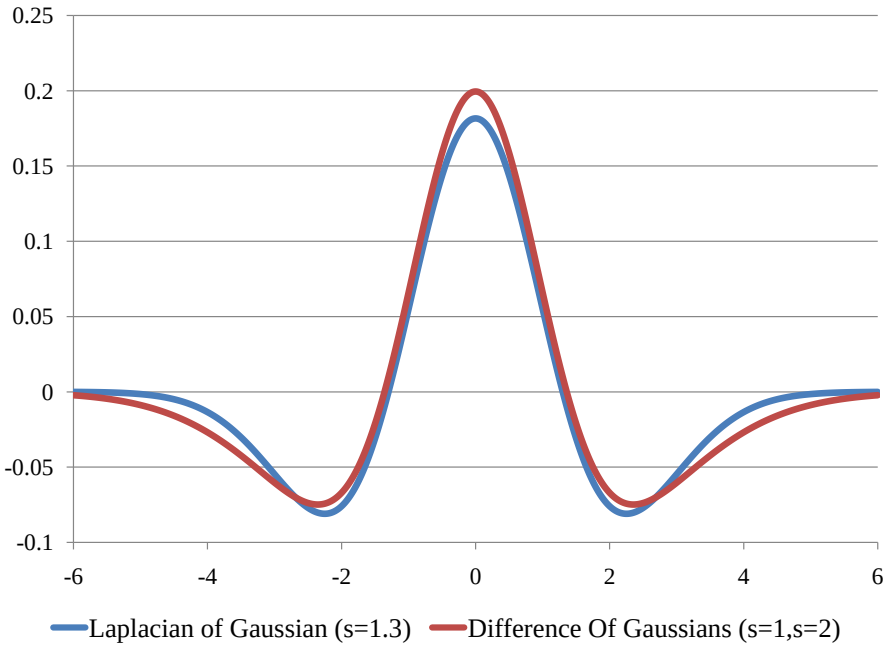


Figure A.6: *Difference of Gaussians* Here we see how the difference of two Gaussians can approximate the shape of a Laplacian of a Gaussian.)

of a Laplacian of a Gaussian over scales computed at the detected point, with the scale being set as the maximum or minimum of the Laplacian over scale.

A.2.3 Blobs

A "blob" is an uniform spot in an image which is either lighter or darker than its surrounds, are are essentially the maxima and minima of the image when convolved with the Laplacian of a Gaussian. The Difference of Gaussian (DoG) operator can be used as an approximation to the Laplacian of a Gaussian to find zero crossings in the gradient of an image for edge detection. It can also be used to find very stable interest points in the center of stable uniform regions [69], and is considered to be a biologically plausible model for the method used by mammalian brains for this purpose [109]. As shown in Figure A.6, the difference of two Gaussians approximates the Laplacian of a Gaussian, and the response of the operator for an image will have a value of zero at edges. The maxima and minima of the response are

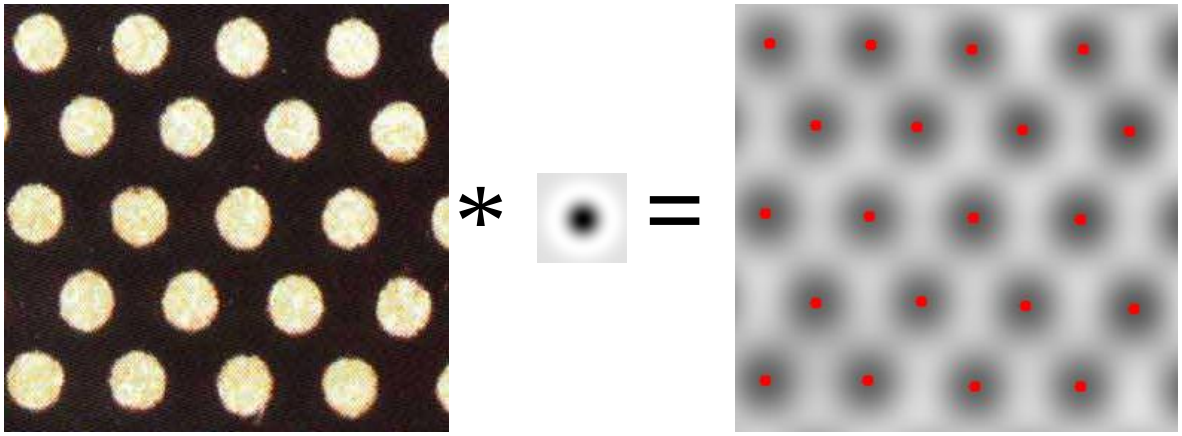


Figure A.7: *Blob Detection* An image is convolved with a Laplacian of Gaussian and blobs are detected as the local minima and maxima of the resulting image. In this figure, the image on the left has been convolved with the Laplacian of a Gaussian shown in the middle to result in the image on the right. The detected blobs are marked with red dots.

stable points of interest. An example of blob detection can be seen in Figure A.7.

Lowe uses the DoG detector to find blobs as the initial step of his effective SIFT feature point scheme [65]. In his system he first calculates a scale-space pyramid by convolving an image with a Gaussian of $\sigma = 2^{\frac{1}{L}}$ L + 3 times and then subsamples the Lth of these images by 2 to begin the next octave. Repeating this O times, this achieves $L + 3$ images in each of O octaves. Within each octave he then calculates $L + 2$ difference images, subtracting the $i + 1$th image from the ith image for $i = 0$ to $i = L + 2$. In each of these images he tests to see if a maximum or minimum in one scale is also the maximum and minimum in the adjacent scales. Once this has been determined to be true, he localizes the point by finding the maximum of a translated Taylor approximation of the scale space function as described in [14] to obtain a series of stable interest points at different scales.

A.3 Descriptors

Once an interesting location in the image has been found, the task remains of describing unique that location in the image. In order to be effective, the description must be robust

to changes in the way the patch looks. These descriptions, called *feature descriptors* are n-tuples which represent information about an area of an image. It is in essence a targeted data reduction which gives particular information about an area in a compact form.

In image matching and retrieval, feature descriptors are calculated using scaled areas around interest points and are utilized to find a correspondence between the points in one image and another. For this purpose, a few properties are desirable:

Perceptually Uniform: Things which "look the same" to a human observer should be near by, while things which "look different" should be far apart.

Robust: Small changes in the following should have little effect:

- Contrast
- Intensity
- Color
- Resolution

Ideally, small location and perspective changes should also have little effect.

Distinctive: Different areas have different descriptors.

Perceptual Uniformity is a term borrowed from color theory, which describes the property that a change in a component value is approximately equally perceptible across the range of that value. In layman's terms, this means that things which "look the same" to a human observer should be near by, while things which "look different" should be far apart.

This is difficult to achieve, and often feature descriptors are compared using Euclidean distance even though the space in which they live is not truly perceptually uniform, with potentially unwelcome effects. In a sense, perceptual uniformity helps to avoid the effects

caused by breaking the triangle inequality. If distance in the space is calculated by using a distance function d, then for feature descriptors A, B, and C

$$d(A, C) \leq d(A, B) + d(B, C) \tag{A.3}$$

If this were not to hold, then it would be possible for A and C to be considered alike but not equally different from B, creating serious problems for comparison.

A *robust* feature descriptor computed for a patch will be the same even if the patch undergoes certain transformations. The image which is "seen" by a camera and then read by a digital computer is dependent upon a large range of variables, not just those of the lens but also lighting conditions in the environment and the encoding method used, among others. A feature descriptor which exhibits large changes for these conditions is thus not useful for matching between scenes represented in different images. The way to achieve this robustness is to utilize interest points in computing the descriptor as opposed to the raw image data. Of particular use is the edge-map of an image, which is robust to changes in contrast, color and intensity. Also, the relative locations of other interest points to the reference point of the descriptor are often used.

In order for a feature descriptor to be *distinctive*, it must be possible for the space in which it lives to be populated uniformly by descriptors computed from different patches. To say it differently, if you chose a random number of image regions and computed the descriptor over those, they would be uniformly distributed over the descriptor space. This property is difficult to achieve, with the usual recourse being to use a large dimensional descriptor.

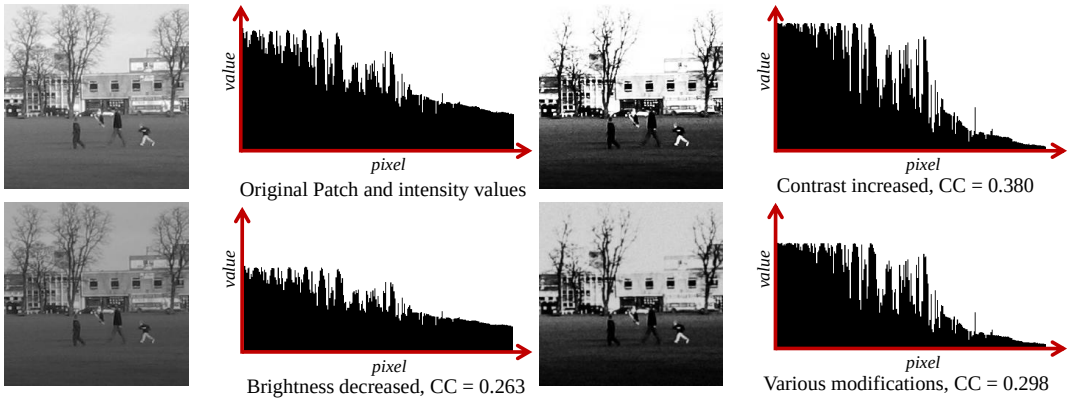


Figure A.8: *Patch Problems*. The values in the histograms are from a lower resolution version of the patches shown, but otherwise are the same intensity values. The histogram is formed by recording the values of each pixel in the patch in row column order. Notice how even small changes in the values of the exact same patch result in completely different pixel values, and thus very low cross-correlation scores.

A.3.1 Zero Normalized Patches

The simplest way to describe an area of an image is to store the intensity values as an array of length N, where N number of pixels in a rectangular patch. It is then possible to compare patches directly using cross-correlation against other patches to find a match.

$$CC(P_1, P_2) = \frac{1}{N} \sum_{i}^{N} P_1[i] P_2[i]. \tag{A.4}$$

This method is not very robust to changes in image conditions, however, as can be seen in Figure A.8. Brightness changes are essentially changes in the mean brightness value over the patch. While the mean changes, the distribution of the intensity values around the mean stays the same. Thus, by giving the intensity values a zero mean, they become immune to brightness change.

$$\mu = \frac{1}{N} \sum_{x,y} I(x, y) \tag{A.5}$$

$$Z(x, y) = I(x, y) - \mu \tag{A.6}$$

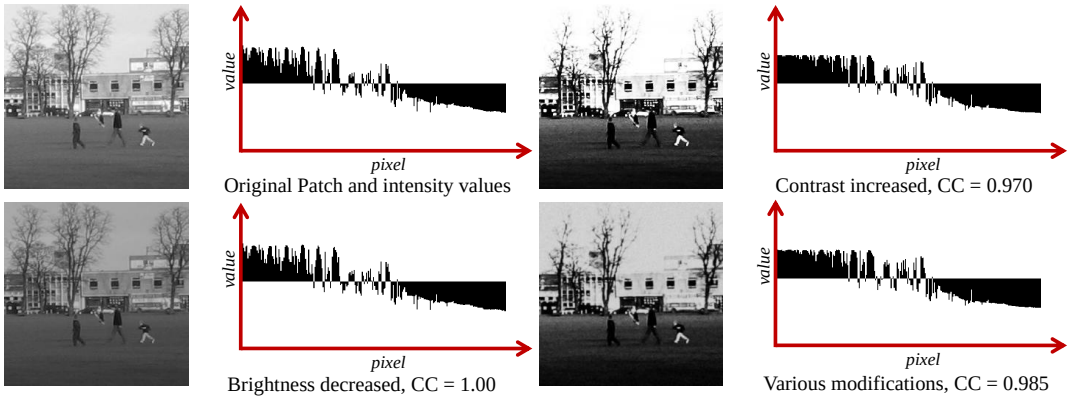


Figure A.9: *Zero-Normalized Cross Correlation*. The values in the histograms are from a lower resolution version of the patches shown, but otherwise are the same intensity values. The histogram is formed by recording the values of each pixel in the patch in row column order, and then transformed to have zero mean and unit variance. By doing this, we remove the effects of the image transformations and receive the correct cross correlation scores.

However, the intensity values are still affected by contrast changes. Contrast is essentially a change in the variance of the distribution of the intensity values around the mean. Thus, to deal with contrast all that is required is to divide each value by the standard deviation of the intensity value distribution.

$$\sigma^2 = \frac{1}{N}\sum_{x,y} Z(x,y)^2$$

$$ZN(x,y) = \frac{Z(x,y)}{\sigma}$$

The resulting collection of zero-mean, unit variance intensity values is known as a zero-normalized patch, and can be accurately matched using simple cross-correlation. The size of the descriptor grows with the size of the patch and the number of color channels, and thus can be quite big. Even so, it is a useful way to represent these areas, as can be seen in Figure A.9.

A.3.2 Filter Banks

The idea of simple image filters being used to describe images, as proposed by Malik and Perona in [66], is motivated by the human vision system. They mention that there exist cells in the eye known as "simple cells" which have receptive fields that are restricted to small regions of space and are highly structured. The simple cell response field takes two forms: oriented or non-oriented, and can be modeled using a two-dimensional filter, as can be seen in Figure 2.5. This was applied to the problem of texture description by Leung and Malik [63], who used the bank of simple filters depicted in Figure A.10. It consists of 8 Laplacian of Gaussian filters and 4 Gaussian filters at different scales to provided non-oriented responses, and 36 oriented filters at 6 different angles, 3 different scales, and 2 different phases. The two phases of oriented filters are first and second derivatives of Gaussians on the minor axis and elongated Gaussians on the major axis, and thus detect edges or bars respectively along their major axes. The descriptor is simply the concatenated responses of all of the filters in the filter bank at a pixel, and is particularly adept at richly encoding texture. Since filter banks respond to basic image features such as blobs, edges, bars, and average brightness, they are innately immune to most changes in an image, as can be seen in Figure A.11. This filter bank is essentially a collection of approximations to Gabor wavelets, and are often referred to as Gabor-like filters. To use wavelets directly, an ideal tool is the dual-tree complex wavelet transform [57] which has been used successfully to describe texture [40; 24].

A.3.3 Orientation Histograms

If you look at every edge in a patch of pixels, you will find that each has a distinct orientation, or way that it is facing. If you look at all of these and weight them by the strength of the edge, you get something that looks like the patch in Figure A.12. These can in turn

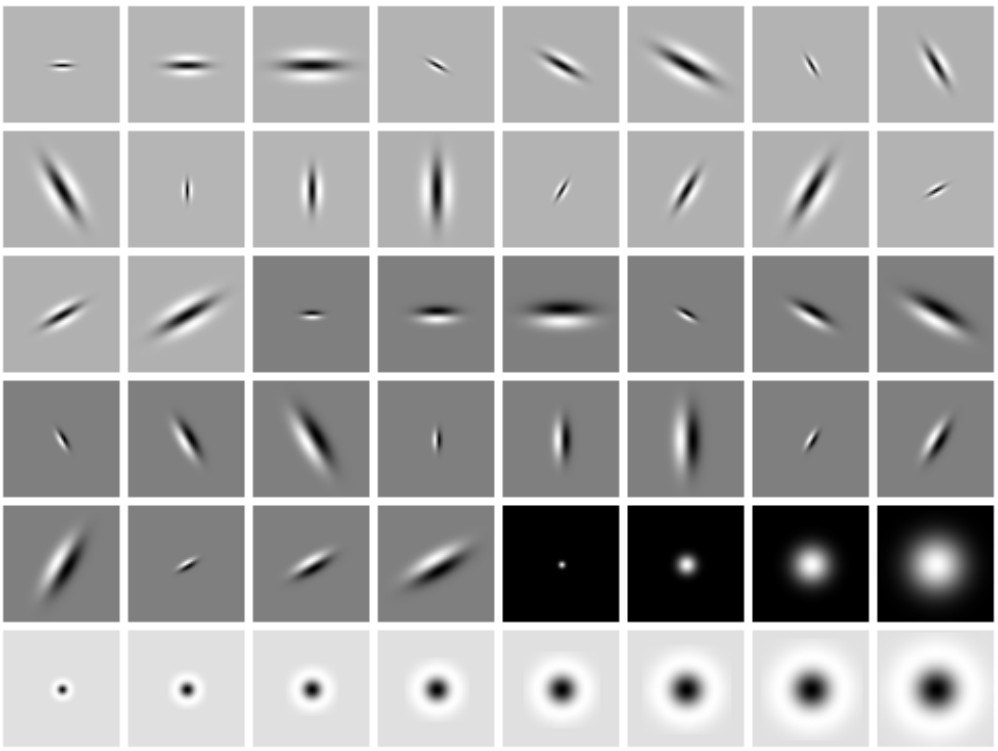


Figure A.10: *A Filter Bank*. The Leung-Malik filter bank, built from simple-cell filters. There are 18 bar filters, 18 edge filters, 4 Gaussian filters and 8 Laplacian of Gaussian filters. See Section A.3.2 for details.

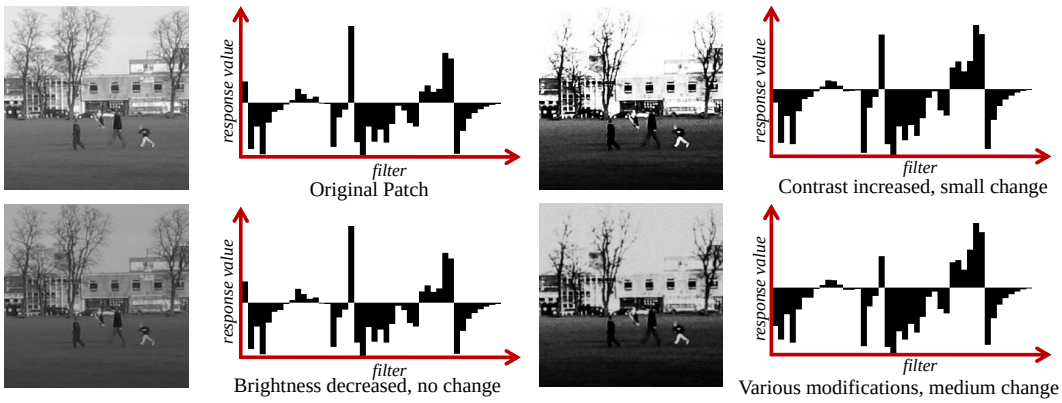
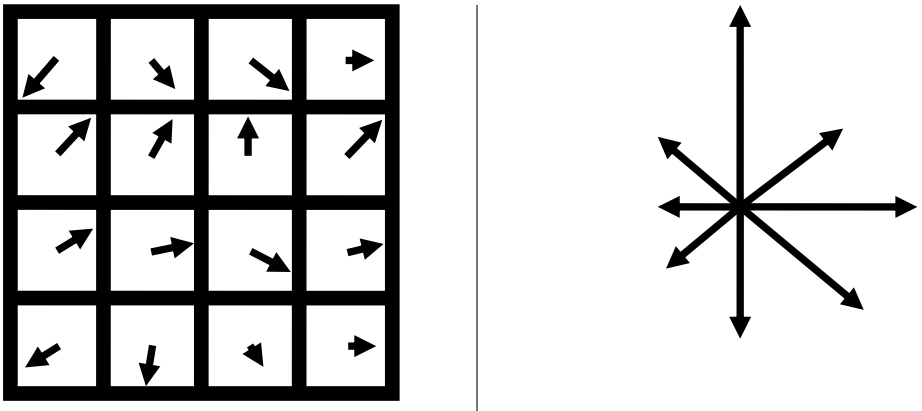


Figure A.11: *Filter Bank Responses*. These vectors are formed by concatenating the responses from the 48 filters in the Leung-Malik filter bank. Here are shown the responses for patches with various transformations. Note the small change in the response values.



(a) (b)

Figure A.12: *Orientation Histograms*. The edges at every pixel in a patch can be collected over the gradient grid (a) into an orientation histogram (b), which represents how many edges point in a particular orientation over the entire grid. The orientation values are interpolated over the direction bins, and its magnitude dictates the amount which each edge contributes to a bin.

be binned together into an orientation histogram. Since this histogram is built using edges, which are robust to contrast and brightness changes and can be detected at different scales, and also incorporates orientation data (thus adding robustness to orientation) this makes them a very strong candidate for a descriptor.

Linear interpolation is used to help alleviate the effects of binning in histograms. Since the angles of edges are continues and the bins of the orientation histogram discrete, we must have a way of "spreading" the value of an edge between the two bins it falls between. We do this by looking at the difference, dx, between a value x and the integer value just before it, i. We distribute a value v which between the two bins i and j in the following way:

$$O_i = O_i + (1 - dx) \times v$$

$$O_j = O_j + dx \times v$$

This results in a more accurate and stable histogram sampling.

The direction of the edge will always be the same regardless of the strength of the edge. Thus, edges which are detectable always end up in the same bins. In order to deal with

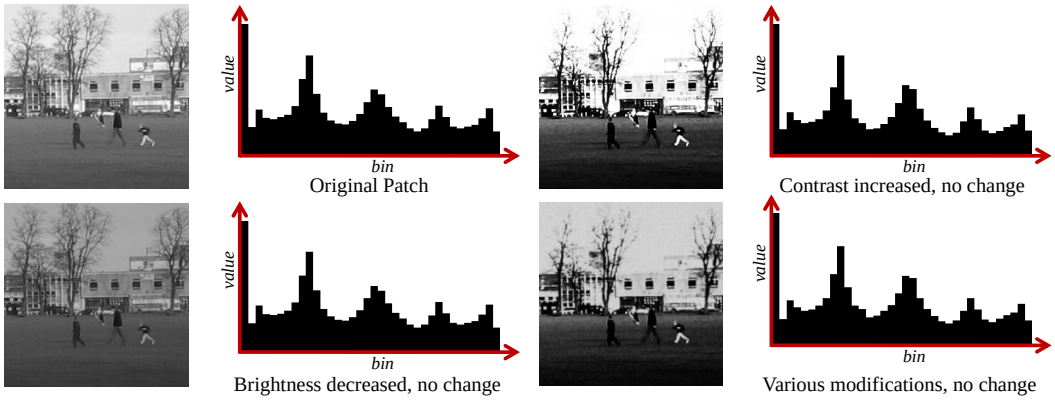


Figure A.13: *Orientation Histograms Responses*. Shown here are orientation histograms with 36 directional bins for various image patches. The edge values are interpolated over the bins and the histograms are normalized to add stability to the final vector. Note the stability of the descriptor over these transformations.

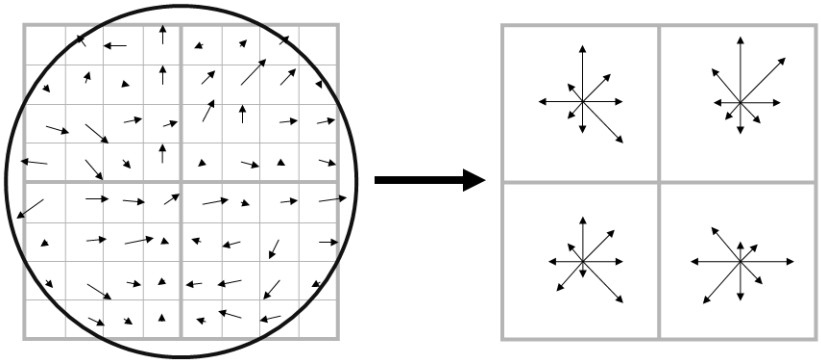


Figure A.14: *The SIFT Descriptor*. The SIFT descriptor is a weighted, interpolated grid of orientation histograms.

changes in edge intensity, the descriptor is normalized. This provides a very stable descriptor, as can be seen in Figure A.13.

A.3.4 SIFT

SIFT stands for **S**cale-**I**nvariant **F**eature **T**ransform, developed by Lowe [65]. It uses a collection of orientation histograms to create a robust and descriptive representation of a patch. This $N \times N$ patch is extracted at the scale of the interest point. Thus, while the patch size never changes the area of the actual image it represents changes depending on scale.

The $N \times N$ patch is split into c cells, and within each cell the intensity gradient at every

pixel is calculated and the directions binned into a histogram weighted by their magnitude and a Gaussian window with a σ of .5 times the scale of the feature centered on the patch. This weights the inner pixels (those closer to the interest point) to avoid possible occlusion problems. A representation of this can be seen in Figure A.14.

If the bins are centered on d directions in each of c cells, the resulting descriptor is a $d \times c$ vector. By dividing the patch into cells, a particular gradient can move around to some degree within the descriptor window and still contribute to the same directional histogram. Once the $d \times c$ vector has been extracted, it is normalized to provide invariance to gradient magnitude change. One final step is performed to help minimize the effects of non-affine lighting changes by thresholding all values in the unit vector to .2 and then renormalizing.

To provide a smoother transition when a gradient moves from one histogram to the next, however, bi-linear interpolation is used to spread its contribution to the nearest cells. Once within the cell, linear interpolation is used to distribute the value to the bins in the orientation histogram.

A.4 Conclusion

While this appendix has covered many different foundational concepts in image processing and computer vision, there is a wealth of information both expanding on these concepts and on others tangental to the research in this thesis but vital for other areas of the field. Cipolla and Giblin provide an excellent introduction to structure from motion and tracking in [19], which uses the interest points and descriptors in this appendix to recover 3D geometry from image sequences, specifically contour. The seminal book on the subject is arguably that of David Marr [68], which revolutionized the field and laid the foundation for many of the advances that were to follow. A good comprehensive guide to the field is provided by Forsyth and Ponce in [34], and can provide a good branching off point into various sub-

specialties.

APPENDIX B

DATASETS

Due to the nature of the problem at hand, specifically the segmentation of images into semantically consistent regions, the data requirement is particularly precise. There are very few datasets currently available which provide such a segmentation. Encouragingly, there has been a growth in the voluntary semantic labeling of images on the internet through such sites as LabelMe [85], innovative games such as *Peekaboom* [105] in which human computation is harnessed for the labeling of images, and the growth of the new segmentation portion for the Pascal Visual Object Challenge [28]. We have used one standard dataset, that being the MSRC21 object recognition and segmentation database introduced in by Shotton *et al*. [92], and one newly adapted dataset. The Spatial Envelope dataset was first introduced by Oliva and Torralba in [76] for use in scene recognition. We refer to it as the "scenes" dataset, and is a good dataset for categorization. In the time since it was made available, it has been added to LabelMe. As we wanted to use it to compare categorization performance and for segmentation, we have written a tool for creating ground truth label images based upon LabelMe annotations, doing automatic translations of synonyms into equivalency classes, of which the most common word is chosen.

Label	Training #	Validation #	Test #
0. void	271	58	253
1. building	68	16	69
2. grass	95	27	91
3. tree	70	11	53
4. cow	23	3	19
5. sheep	15	4	16
6. sky	75	15	68
7. airplane	14	3	13
8. water	36	6	33
9. face	27	6	27
10. car	22	3	19
11. bicycle	15	3	14
12. flower	17	3	15
13. sign	15	3	13
14. bird	18	3	17
15. book	17	3	15
16. chair	14	3	13
17. road	70	17	64
18. cat	11	3	10
19. dog	14	3	13
20. body	31	7	30
21. boat	15	3	15
Total	276	59	256

Table B.1: *MSRC21 Image Counts* There are 591 images total, almost all images have more than one category present. Counts are of images in the training, validation, or test set which contain pixels with the row's label.

Label	Training #	Validation #	Test #
0. void	29.64%	26.16%	27.46%
1. building	7.59%	7.71%	7.51%
2. grass	13.38%	18.60%	14.38%
3. tree	6.39%	7.16%	6.08%
4. cow	2.24%	1.02%	2.10%
5. sheep	1.54%	1.59%	1.67%
6. sky	6.70%	6.01%	7.11%
7. aeroplane	1.11%	0.92%	0.92%
8. water	5.84%	3.27%	5.67%
9. face	1.27%	1.05%	1.28%
10. car	2.29%	2.14%	2.47%
11. bicycle	1.96%	2.05%	1.80%
12. flower	1.83%	2.34%	2.57%
13. sign	1.34%	1.59%	2.20%
14. bird	1.02%	0.51%	0.94%
15. book	3.71%	3.87%	3.85%
16. chair	1.22%	1.37%	1.42%
17. road	6.55%	7.04%	5.91%
18. cat	1.23%	1.21%	1.01%
19. dog	1.05%	1.15%	1.51%
20. body	1.62%	2.02%	1.41%
21. boat	0.47%	1.22%	0.74%
Pixel Total	19242560	4125120	17812160

Table B.2: *MSRC21 Pixel Percentages* Percentages are out of the total pixel counts in the bottom row.

Figure B.1: *Sample MSRC21 Images and Ground Truth.*

Label	Training #	Validation #	Test #
0. void	486	45	2199
1. water	157	9	511
2. sky	422	41	1879
3. mountain	179	14	850
4. tree	234	22	1085
5. grass	100	6	150
6. road	181	23	691
7. sidewalk	100	16	358
8. building	227	25	1041
9. rock	100	5	169
10. snow	31	3	35
11. sand	95	3	99
12. plant	100	8	281
13. car	131	19	515
14. sign	100	10	236
15. person	100	13	238
Total	486	45	2202

Table B.3: *Scenes Image Counts*. There are 2742 images total, almost all images have more than one category present. Counts are of images in the training, validation, or test set which contain pixels with the row's label.

B.1 MSRC21 Dataset

The MSRC21 dataset was first used in [92], and consists of 591 ground-truth labeled images consisting of 21 categories + a void/background category. The names of the categories and image counts are in Table B.1 and percentages of total pixels per category are in Table B.2. Sample images with their accompanying ground truth labeling are shown in Figure B.1.

B.2 Scenes Dataset

The Scenes database was acquired from the LabelMe website, after which it was automatically processed by our conversion software. The large number of sometimes ambiguous or contradictory labels given by the anonymous users were collapsed into the top occurring words where possible using WordNet synonym sets [71]. Unrecognized words were then manually mapped to the largest 18 labels or to an ignore label. The annotations were then

Figure B.2: *Sample Scenes Images and Ground Truth.*

Label	Training #	Validation #	Test #
0. void	7.04 %	5.96 %	9.42 %
1. water	9.30 %	7.27 %	6.25 %
2. sky	24.92 %	22.05 %	24.79 %
3. mountain	6.94 %	9.13 %	12.83 %
4. tree	9.78 %	9.75 %	10.82 %
5. grass	4.14 %	2.47 %	1.43 %
6. road	8.09 %	11.01 %	6.82 %
7. sidewalk	1.15 %	1.20 %	0.83 %
8. building	17.85 %	21.65 %	21.27 %
9. rock	1.79 %	1.02 %	0.79 %
10. snow	1.07 %	1.54 %	0.29 %
11. sand	4.05 %	1.84 %	1.13 %
12. plant	2.12 %	2.41 %	1.90 %
13. car	1.25 %	2.10 %	1.16 %
14. sign	0.29 %	0.31 %	0.15 %
15. person	0.23 %	0.29 %	0.13 %
Pixel Total	31850496	2949120	144310272

Table B.4: *Scenes Pixel Percentages* Percentages are out of the total pixel counts in the bottom row.

combined to create ground truth images of the kind seen in Figure B.2. These ground truth images were then used to create a balanced training/validation/test split, the counts and percentages of which are shown in Figures B.3 and B.4.

In addition to the ground truth labels, each image has an additional scene label from the dataset's original use for image categorization. It is this additional label which is used for categorization in Chapters 2 and 3.

APPENDIX C

EXPLANATION OF ATTACHED CD-ROM

The accompanying CD-ROM contains a demonstration copy of the Palette Search software and additional segmentations to those already presented in the text. This appendix describes how to install the Palette Search software, and the layout of the folders containing various segmentations.

C.1 Installing the Palette Search Demonstration Software

In the 'paletteSearch' folder on the CD-ROM is found the executable 'setup.exe'. The demo runs on the Microsoft Windows© operation system, and requires the following support software:

- Windows Installer 3.1 (or higher)
- .NET Framework 3.5 (or higher)

which must be obtained directly from Microsoft. If this software is present, then executing 'setup.exe' will install the demonstration software. The software is described sufficiently in Chapter 7, but if any further help is needed a tutorial is included as part of the program. Simply press the 'Tutorial' button to begin.

C.2 MSRC21 Results

The MSRC21 segmentation results are located in the 'msrc21' folder. We include three sets of results, the raw STF semantic segmentations from Chapter 3 in subfolder 'stf', the full semantic segmentations of Chapter 4 in folder 'segmentation', and the segmentation results from Chapter 5 in folder 'annotation'. The ground truth images are located in the 'groundTruth' folder, and the dataset images in the 'images' folder. Webpages for ease of viewing are provided in the base directory, named in the same manner as the folders.

C.3 Scenes Results

The Scenes segmentation results are located in the 'msrc21' folder. We include three sets of results, the raw STF semantic segmentations from Chapter 3 in subfolder 'stf', the full semantic segmentations of Chapter 4 in folder 'segmentation', and the segmentation results from Chapter 5 in folder 'annotation'. The ground truth images are located in the 'groundTruth' folder, and the dataset images in the 'images' folder. Webpages for ease of viewing are provided in the base directory, named in the same manner as the folders.

APPENDIX D

PUBLICATIONS

The author's publications to date and how they relate to this thesis are as follows:

Cross Modal Disambiguation *(with K. Barnard)* [5]

Word Sense Disambiguation with Pictures *(with K. Barnard)* [4]

In this work, we used machine learning techniques from computational linguistics and computer vision to leverage visual data for the task of word sense disambiguation in accompanying text. Many words have several meanings, or are polysemous, and combinations of polysemous words in a document can result in several conflicting interpretations of it. This complicates the task of document categorization and tagging, and is a very difficult problem to solve when the only source of information is the text itself. Accompanying photographs, such as those for a newspaper article, can be used to disambiguate words in the document, making this categorization task easier. This work began the author's interest in understanding the meaning of images, and in the linking of images to semantics.

Improved Image Annotation and Labeling Through Multi-Label Boosting [55]

This paper concentrated on using boosting techniques to improve existing multi-label image annotation and labeling systems. The concentration here was on labeling areas of the image with semantic tags and providing keyword annotations for the global image, much as is shown here in Chapter 5.

Semantic Photo Synthesis *(with J. Shotton, G. Brostow, V. Kwatra)*[53]

This paper introduced the concept of searching using semantic paint that was the basis of the system in Chapter 7. That work also used a database of semantically labeled images and performed spatial searches, though both systems were not as developed due to the concentration on finding candidate images for synthesis of user queries. It was introduced in an oral presentation at the Eurographics conference in Vienna in 2006 (reference above), and was the subject of an invited talk at HVEI in 2007 [54].

Semantic Texton Forests *(with J. Shotton)*[91]

This is our most recent work, a digest version of Chapters 3 and 4. It was submitted and accepted as an oral presentation to CVPR in 2008.

BIBLIOGRAPHY

[1] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997. 31

[2] K. Barnard, P. Duygulu, N. de Freitas, D. Forsyth, D. Blei, and M. I. Jordan. Matching words and pictures. *J. of Machine Learning Research*, 3:1107–1135, 2003. 10, 17, 73, 77, 97

[3] K. Barnard and P. Gabbur. Color and color constancy in a translation model for object recognition. In *Color Image Conference*, pages 364–369, 2003. 15

[4] K. Barnard and M. Johnson. Word sense disambiguation with pictures. *Artificial Intelligence*, 167:13–30, 2005. 73, 97, 153

[5] K. Barnard, K. Yanai, M. Johnson, and P. Gabbur. Cross modal disambiguation. In *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*, pages 248–259. Springer-Verlag, 2006. 97, 153

[6] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(24):509–522, April 2002. 20, 22

[7] T. Berners-Lee and M. Fischetti. *Weaving the Web*. Harper, San Francisco, 1999. 1

[8] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006. 54, 55, 74, 75, 84

[9] D. M. Blei. *Probabilistic models of text and images*. PhD thesis, University of Berkeley, 2004. 73

[10] A. Bosch, A. Zisserman, and X. Mu noz. Representing shape with a spatial pyramid kernel. In *Proc. Int. Conf. on Image and Video Retrieval*, 2007. 8

[11] A. Bosch, A. Zissermann, and X. Muñoz. Image classification using random forests and ferns. In *Proc. Int. Conf. on Computer Vision*, 2007. 33

[12] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 29, 31

[13] L. Breiman, J.H. Friedman, and R. Olshen. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984. 31

[14] M. Brown and D. G. Lowe. Invariant features from interest point groups. In *Proc. British Machine Vision Conference*, pages 656–665, Cardiff, Wales, 2002. 133

[15] J. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986. 19, 129

[16] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(3):394–410, 2006. 96

[17] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *Proceedings of VISUAL 1999*, pages 509–516, 1999. 10

[18] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. 23

[19] R. Cipolla and P. J. Giblin. *Visual Motion of Curves and Surfaces*. Cambridge University Press, Cambridge, 2000. 142

[20] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas, and P. N. Vianilos. The bayesian image retrieval system, *PicHunter*: Theory, implementation and psychological experiments. *IEEE Trans. on Image Processing*, 9(1):20–37, 2000. 98

[21] N. Cristianini, H. Shawe-Taylor, and H. Lodhi. Latent semantic kernels. *J. of Intelligent Information Systems*, 18(2-3):127–152, 2002. 11

[22] G. Csurka, C.R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004. 10, 11, 15

[23] J.G. Daugman. Two-dimensional spectral analysis of cortical receptive field profiles. *Vision Res.*, 20(10):847–856, 1980. 17

[24] P. F. C. de Rivaz and N. G. Kingsbury. Complex wavelet features for fast texture image retrieval. In *Proc. IEEE Conf. on Image Proc.*, Kobe, Japan, October 1999. 138

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. 77

[26] P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In A. Heyden, G. Sparr, and P. Johansen, editors, *Proc. European Conf. on Computer Vision*, volume LNCS 2353, pages 97–112. Springer, May 2002. 73, 97

[27] J. Eakins. Automatic image content retrieval - are we getting anywhere? In *Proc. of Int. Conf. on Electronic Library and Visual Information Research*, pages 123–135, May 1996. 87

[28] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL VOC Challenge 2007. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 123, 144

[29] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005. 10, 15, 58

[30] L. Feigenbaum, I. Herman, T. Hongsermeier, E. Neumann, and S. Stephens. The semantic web in action. *Scientific American*, 297(6):90–97, 2007. 2

[31] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient belief propagation for early vision. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 261–268, June 2004. 9, 17

[32] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *Proc. Int. Conf. on Computer Vision*, volume 2, pages 1816–1823, Beijing, China, October 2005. 12

[33] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, and B. Dom. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, 1995. 96

[34] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, August 2002. 142

[35] B. J. Frey and D. J. C. MacKay. A revolution: Belief propagation in graphs with cycles. In *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998. 78, 91, 92

[36] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 36(1):3–42, 2006. 29, 31

[37] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proc. Int. Conf. on Computer Vision*, 2005. 50, 59

[38] V. N. Gudivada and V. V. Raghavan. Content-based image retrieval systems. *IEEE Computer*, 28(9):18–22, 1995. 87

[39] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, Manchester, 1988. 130

[40] S. Hatipoglu, S. K. Mitra, and N. G. Kingsbury. Texture classification using dual-tree complex wavelet transform. In *Proc. Int. IEE Conference on Image Processing and Its Applications*, pages 344–347, Manchester, England, July 1999. 138

[41] X. He, R.S. Zemel, and M.Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 695–702, June 2004. 10

[42] K.A. Heller and Z. Ghahramani. A simple bayesian framework for content-based image retrieval. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006. 97

[43] R. Hirsch. *Exploring Colour Photography: A Complete Guide*. Laurence King, London, UK, 2004. 126

[44] T. Hofmann. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In *Proc. of the Intl. Joint Conference in Artificial Intelligence*, 1999. 73

[45] S. C. H. Hoi and M. R. Lyu. A semi-supervised active learning framework for image retrieval. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005. 98

[46] S. Holtzman. *Intelligent Decision Systems*. Addison-Wesley, 1989. 31

[47] R. W. G. Hunt. *The Reproduction of Colour, 6th ed.* Wiley, Chichester, UK, 2004. 126

[48] R. S. Hunter. Accuracy, precision, and stability of new photo-electric color-difference meter. *J. of the Optical Society of America*, 38(12), December 1948. 17, 128

[49] R. S. Hunter. Photoelectric color-difference meter. *J. of the Optical Society of America*, 38(7), July 1948. 17, 128

[50] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, New Jersey, USA, 1989. 9, 44, 128, 129

[51] I.H. Jermyn, C. Shaffrey, and N. Kingsbury. The methodology and practice of the evaluation of image retrieval systems and segmentation methods toutes les pages. In *INRIA*, 2003. 121

[52] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proc. European Conf. on Machine Learning*, number 1398 in 10, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE. 11

[53] M. Johnson, G. Brostow, J. Shotton, O. Arandjelovic, V. Kwatra, and R. Cipolla. Semantic photo synthesis. *Computer Graphics Forum*, 25(3):407–413, September 2006. 88, 98, 154

[54] M. Johnson, G. Brostow, J. Shotton, V. Kwatra, and R. Cipolla. Semantic photo synthesis. In *SPIE: Human Vision and Electronic Imaging XII*, San Jose, USA, March 2007. 154

[55] M. Johnson and R. Cipolla. Improved image annotation and labeling through multi-label boosting. In *Proc. British Machine Vision Conference*, September 2005. 153

[56] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, March 1981. 17

[57] N.G. Kingsbury. Complex wavelets for shift invariant analysis and filtering of signals. *J. Applied and Computational Harmonic Analysis*, 10(3):234–253, May 2001. 138

[58] F. R. Kschischnang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory*, 47(2):498–519, 2001. 78, 91, 92

[59] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. Int. Conf. on Machine Learning*, pages 282–289, 2001. 84, 122

[60] I. Laptev. Improvements of object detection using boosted histograms. In *Proc. British Machine Vision Conference*, volume III, pages 949–958, Edinburgh, UK, 2006. 40

[61] J. Lasserre, A. Kannan, and J. Winn. Hybrid learning of large jigsaws. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Minneapolis, 2007. 56

[62] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 2:775–781, 2005. 33, 38, 40

[63] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int. J. Computer Vision*, 43(1):29–44, June 2001. 17, 138

[64] H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. J. C. H. Watkins. Text classification using string kernels. In *NIPS*, pages 563–569, 2000. 11

[65] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Computer Vision*, 60(2):91–110, 2004. 14, 15, 133, 141

[66] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *J. Optical Society of America A*, 12(7):629–639, July 1990. 17, 129, 138

[67] R. Marée, P. Geurts, J. Piater, and L. Wehenkel. Random subwindows for robust image classification. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 34–40, June 2005. 33

[68] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY, USA, 1982. 142

[69] K. Mikolajczyk. *Detection of local features invariant to affine transformations*. PhD thesis, Institut National Polytechnique de Grenoble, France, 2002. 131, 132

[70] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2003. 9, 14, 15, 20

[71] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. Introduction to WordNet: an online lexical database. *International Journal of Lexicography*, 3(4):235–244, 1990. 148

[72] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *NIPS*, 2006. 33, 44, 50

[73] H. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical Report CMU-RI-TR-3, Carnegie-Mellon University, September 1980. 130

[74] J. A. Noble. *Descriptions of Image Surfaces*. PhD thesis, University of Oxford, 1989. 131

[75] M. O'Keefe. New infotrends study indicates continued growth in photo imaging services, May 2007. 3

[76] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int. J. Computer Vision*, 42(3):145–175, 2001. 22, 69, 144

[77] A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. *Visual Perception, Progress in Brain Research*, 155(1):23–26, 2006. 10, 58

[78] E. Panofsky. *Meaning in the visual arts*. Doubleday Anchor Books, Garden City, NY, 1955. 88

[79] P. Quelhas, F. Monay, J.-M. Odobez, D. Gatica, and T. Tuytelaars. Modeling scenes with local descriptors and latent aspects. In *Proc. Int. Conf. on Computer Vision*, 2005. 10, 12, 15

[80] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *Proc. Int. Conf. on Computer Vision*, 2007. 63

[81] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proc. Int. Conf. on Computer Vision*, volume 1, pages 10–17, Nice, 2003. 10

[82] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Computer Vision*, 40(2):99–121, November 2000. 98

[83] B. C. Russell, A. Torralba, C. Liu, R. Fergus, and W. T. Freeman. Object recognition by scene alignment. In *NIPS*, 2007. 58

[84] B.C. Russell, A.A. Efros, J. Sivic, W.T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 1605–1614, June 2006. 15, 123

[85] B.C. Russell, A. Torralba, K. Murphy, and W. T. Freeman. *LabelMe*: a database and web-based tool for image annotation. *Int. J. Computer Vision*, 77:157–173, 2008. 3, 69, 123, 144

[86] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988. 24

[87] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambrige, MA, 2002. 23

[88] N. Shadbolt, T. Berners-Lee, and W. Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006. 1

[89] Cián Shaffrey. *Multiscale Techniques for Image Segmentation, Classification and Retrieval*. PhD thesis, University of Cambridge, 2003. 96

[90] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Seattle, June 1994. 131

[91] J. Shotton and M. Johnson. Semantic texton forests. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008. 154

[92] J. Shotton, J. Winn, C. Rother, and A. Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Proc. European Conf. on Computer Vision*, volume LNCS 3951, pages 1–15. Springer, May 2006. 9, 10, 23, 30, 38, 63, 64, 68, 70, 84, 120, 144, 148

[93] J. Sivic, B.C. Russell, A.A. Efros, A. Zisserman, and W.T. Freeman. Discovering objects and their localization in images. In *Proc. Int. Conf. on Computer Vision*, volume 1, pages 370–377, Beijing, China, October 2005. 10, 15, 73

[94] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. Int. Conf. on Computer Vision*, pages 2: 1470–1477, 2003. 11, 15

[95] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000. 96, 107, 121

[96] A. R. Smith. Color gamut transform pairs. *Computer Graphics*, 12(3):12–19, August 1978. 128

[97] M. Swain and D. Ballard. Color indexing. *Int. J. Computer Vision*, 7:11–32, 1991. 50

[98] V. Uren, Y. Lei, V. Lopez, H. Liue, E. Motta, and M. Giordanino. The usability of semantic search tools: a review. *Knowledge Engineering Review*, 22(4):361–377, 2007. 7, 95

[99] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Proc. Int. Conf. on Computer Vision*, Rio de Janeiro, Brazil, 2007. 8

[100] N. Vasconcelos. Minimum probability of error image retrieval. *IEEE Trans. Signal Processing*, 52(8):2322–2336, 2004. 96, 97

[101] N. Vasconcelos. From pixels to semantic spaces: Advances in content-based image retrieval. *IEEE Computer*, 40(7):20–26, July 2007. 7, 97

[102] N. Vasconcelos and A. Lippman. A bayesian framework for content-based indexing and retrieval. In *Proc. of IEEE Data Compression Conference*, 1998. 97

[103] J. Verbeek and B. Triggs. Region classification with markov field aspect models. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007. 68, 70, 120

[104] P. Viola and M.J. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 511–518, December 2001. 30, 38

[105] L. von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. In *Proc. of SIGCHI conf. on Human Factors in Computing Systems*, pages 55–64, Montréal, Québec, Canada, 2006. 144

[106] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Proc. Int. Conf. on Computer Vision*, volume 2, pages 1800–1807, Beijing, China, October 2005. 10, 58

[107] A. Yavlinsky, E. Schofield, and S. Rüger. Automated image annotation using global features and robust nonparametric density estimation. In *Proc. Int. Conf. on Image and Video Retrieval*, 2005. 97

[108] R. M. Yerkes. *Introduction to Psychology*. Henry Holt and Co., New York, 1911. 128

[109] R. A. Young, R. M. Lesperance, and W. W. Meyer. The gaussian derivative model for spatial-temporal vision: I. corticla model. *Spatial Vision*, 14(3,4):261–319, 2001. 132

[110] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classificaiton of texture and object categories: A comprehensive study. *Int. J. Computer Vision*, 73(2):213–238, 2007. 15, 22, 59

[111] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data*, 2003. 98