Discriminative Techniques For The Recognition Of Complex-Shaped Objects

Owen Carmichael

September 18, 2003

CMU-RI-TR-03-34

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:

Martial Hebert, Chair
Henry Schneiderman
Tom Minka, Statistics Department
Jianbo Shi, University of Pennsylvania
Yann LeCun, New York University

This research was supported in part by grants from the National Science Foundation and the Honda Corporation.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of Carnegie Mellon University.

Keywords: Computer Vision, Object Recognition, Machine Learning, Shape, Texture.

Abstract

This thesis presents new techniques which enable the automatic recognition of everyday objects like chairs and ladders in images of highly cluttered scenes. Given an image, we extract information about the shape and texture properties present in small patches of the image and use that information to identify parts of the objects we are interested in. We then assemble those parts into overall hypotheses about what objects are present in the image, and where they are. Solving this problem in a general setting is one of the central problems in computer vision, as doing so would have an immediate impact on a far-reaching set of applications in medicine, surveillance, manufacturing, robotics, and other areas.

The central theme of this work is that formulating object recognition as a discrimination problem can ease the burden of system design. In particular, we show that thinking of recognition in terms of discriminating between objects and clutter, rather than separately modeling the appearances of objects and clutter, can simplify the processes of extracting information from the image and identifying which parts of the image correspond with parts of objects.

The bulk of this thesis is concerned with recognizing "wiry" objects in highly-cluttered images; an example problem is finding ladders in images of a messy warehouse space. Wiry objects are distinguished by a prevalence of very thin, elongated, stick-like components; examples include tables, chairs, bicycles, and desk lamps. They are difficult to recognize because they tend to lack distinctive color or texture characteristics and their appearance is not easy to describe succinctly in terms of rectangular patches of image pixels. Here, we present a set of algorithms which extends current capabilities to find wiry objects in highly cluttered images across changes in the clutter and object pose. Specifically, we present discrimination-centered techniques for extracting shape features from portions of images, classifying those features as belonging to an object of interest or not, and aggregating found object parts together into overall instances of objects. Moreover, we present a suite of experiments on real, wiry objects– a chair, cart, ladder, and stool respectively– which substantiates the utility of these methods and explores their behavior.

The second part of the thesis presents a technique for extracting texture features from images in such a way that features from objects of interest are both well-clustered with each other and well-separated from the features from clutter. We present an optimization framework for automatically combining existing texture features into features that discriminate well, thus simplifying the process of tuning the parameters of the feature extraction process. This approach is substantiated in recognition experiments on real objects in real, cluttered images.

Acknowledgements

For their tireless love and support, I thank Gretchen, Mom, Dad, Cathy, and Lesley. Without their patience and care during difficult periods, I would never have finished graduate school or much else. Also, I thank the Reinhart family for their unending generosity and for providing me with a loving second home in Pittsburgh. For the companionship and good times that kept my spirits up throughout graduate school and life, I need to thank all my friends, especially my Robotics mates Nick Roy , Sal Desiano, Pragyan Mishra, Mike Montemerlo, Bart Nabbe, Danny Huber, Dave Tolliver, Jake Sprouse, and everyone else on the second floor of Smith Hall; The P List, The Brothas, Jim, Ed, Kelly, my Berkeley cronies, and everyone else from home; WRCT people, frisbee people and anyone else I forgot in Pittsburgh.

For many years of priceless technical advice and wisdom which shaped my development as a researcher, I thank Martial Hebert. Additionally, I thank my committee members, Henry Schneiderman, Tom Minka, Jianbo Shi, and Yann LeCun for taking the time to provide helpful comments and advice during this project. Other people whose feedback proved invaluable throughout the course of my thesis work include Danny Huber, Shyjan Mahamud, Takeo Kanade, Chuck Rosenberg, Bill Freeman, Dave Tolliver, Pragyan Mishra, the members of the MISC reading group, and members of VASC at large.

For various forms of help, advice, and support over the years, thanks go to Jim Osborn, Scott Thayer, Eiro Fujii and all the members of his group at Minolta Reserach Lab. For crucial technical support which got everything back up and running while I am shouting expletives at the computer, I thank Tim Doebler, Bart Nabbe, and Chuck Rosenberg. For funding, I thank NSF, Honda, Kodak, Minolta, and the DOE.

Finally, for taking the time to believe in me and investing their minds and hearts into teaching me how to be a better person, I thank the teachers who have shaped my mind and made a profound impact on my development over the years. They include Charles Wohl, Tom Anderson, Marshall Foletta, Dale Zarzana, Tim and Claudine Sweeters, Steve Larimore, Daryl Ross, and Lenzie Williams.

I couldn’t have done this without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is concerned with object recognition, broadly defined as the problem of detecting the presence and characteristics of physical objects in still photographs or video. Long considered one of the central problem areas in computer vision, object recognition summarizes one important aspect of the long-term goal of the field: the automated, intelligent summary by computers of the contents of images.

In this thesis we present computational techniques for detecting objects in images based on their shape and appearance. Figure 1.1 represents the subject matter of this work pictorially: presented with an image containing objects like ladders, carts, and chairs (left column), we want a computer program to analyze the image and report to us the locations of those objects in the image (right column). We present two techniques here which are designed to be applied to two distinct types of objects. Chapters 3 through 6 focus on techniques for recognizing objects that have limbs, holes, and long, thin components (like those in Figure 1.1). Chapter 7, on the other hand, is more concerned with recognizing objects that are "filled in" like the mug in Figure 1.2.

The main contribution of the thesis is a method for finding "wiry" objects– objects with thin wire-like structures like chair legs– based on the shapes they form in the image. The problem of finding wiry objects in cluttered scenes has challenged object recognition systems in the past, mainly because building representations of how wiry objects appear across a wide range of imaging conditions is difficult. In particular, many successful object recognizers have been built which represent the appearances of objects in terms of collections of one or more large rectangular image patches (for example [104][118]), but the thin, linear components of wiry objects do not usually fill up rectangular regions of the image.

Moreover, in previous attempts at representing objects based on their shape (*e.g.* [64]), the simple geometric characteristics used to describe the objects are difficult to distinguish from the geometric characteristics of clutter objects. Here, our central contribution is a technique for shape-based object recognition which constructs rich, informative image features discriminatively– that is, rather than focus on modeling the appearance of the wiry objects, we concentrate on modeling the differences between the appearance of wiry objects and the appearance of objects in the background. We demonstrate that our approach enables the recognition of complex-shaped objects appearing at arbitrary orientations in complicated scenes.

Over the past forty years, researchers have spent considerable effort exploring various aspects of object recognition. To place this work into the context of the overall field, Section 1.1 develops a taxonomy of object recognition research in terms of four broad criteria, and describes the work in this thesis in terms of the taxonomy. Subsequently, Section 1.2 presents a specific recognition paradigm, called *bottom-up localization*, which we follow in this document, and spells out the contributions we make to the field in terms of the paradigm. Section 1.3 briefly reviews some of the applications of object recognition to real-world problems.

Then, Chapter 2 explores in more detail the prior recognition techniques most closely related to ours. Chapters 3 through 6 deal with a new technique for recognizing wiry objects in images based on shape. The basic algorithm is in Chapter 3; Chapter 4 presents extensions to enhance its usefulness; and 5 reports the results of a set of quantitative experiments evaluating various aspects of our approach. Chapter 6 discusses some of the limitations of the approach, future research directions, and related work. In Chapter 7 we propose a technique for recognizing objects based on their appearance properties, and show a set of experiments which validate its usefulness.

1.1 A Taxonomy Of Object Recognition Research

A large body of work has been published on the overall problem of detecting and describing objects in images. A panoply of objects, from car parts to skyscrapers, have been detected, and a wide variety of symbolic and numerical computation techniques have been employed to reason about how those objects appear. Unfortunately, however, very little exists in the way of a concise, structured vocabulary according to which previous work on object recognition can be delineated. In this section, we present four basic criteria which we feel

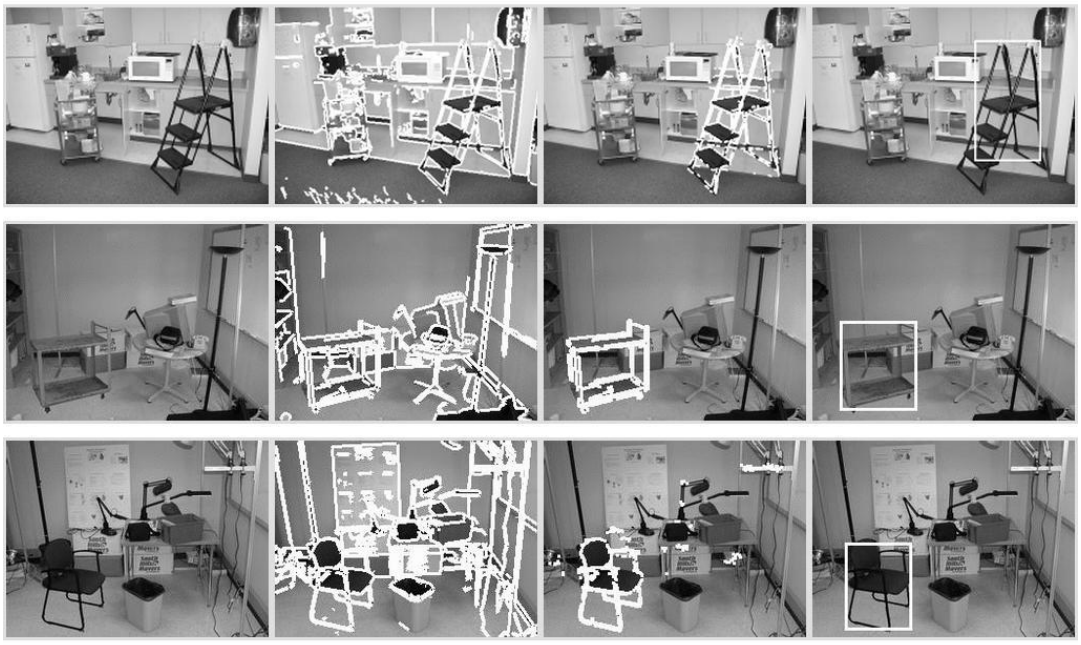


Figure 1.1: This document presents approaches to bottom-up localization, three examples of which are shown in this figure. Left to right: input images, detected edges, part classification results, and aggregation.

capture the most important characteristics of previous object recognition algorithms, and by which previous projects and papers can be the most clearly differentiated. The purpose of this taxonomy is to provide a vocabulary which we employ in Section 1.2 to succinctly describe the contributions of the thesis. Throughout this section, we will use the problem of detecting chairs in images as a motivating example. The four criteria are:

- **Problem Definition.** A problem definition is a detailed description of which exact quantities related to the objects the algorithm is required to estimate. For example, the problem may be to determine whether images contain chairs, and additionally determine what kinds of chairs they are– recliners, office chairs, etc. Or, the problem may be defined as determining whether chairs are present, and what their relative positions are with respect to the camera.

- **Sources of Variability.** Images of chairs taken at different times can look different due to a variety of factors: moving the chairs and other objects around, turning lights on or off, or changing the zoom on the camera can cause an image of a chair to vary dramatically. Each algorithm makes different assumptions about which of these sources of variability– object motion, lighting changes, and so on– are expected to change the appearance of the objects and background over the set of all images.

- **Cues.** Cues are the characteristics of the image which allow us to differentiate the objects from the background. If the problem is to find a particularly brightly-colored chair in an otherwise grey room, color may be a useful cue; likewise the chairs may be identified based on their shape, patterns on the upholstery, the way light shines off them, and so on.

- **Methods.** These are the computational strategies taken in the search for objects in the image. One algorithm may first search the image for sections that look like legs, armrests, and seats, and then reason about the configurations of these parts in order to determine whether chairs are present; another algorithm may first look for large areas of the image which appear vaguely chair-like, then examine each area in more detail for evidence of chair parts.

In this thesis, we address the problem of detecting the presence of objects in images and estimating their spatial extents in the images. The sources of variability we accommodate are the poses of objects and the characteristics of the background of the image. We address the use of shape and visual texture patterns as cues, and we take the former of the two approaches listed under **Methods** above: that is, we first search for object components, and then fuse these components into overall objects. The following sections describe each of these criteria in more detail and give examples of how our work and some of the more important prior approaches to object recognition address them. Our categorization of this document and many previous papers in terms of the four criteria is shown in Table 1.1.

1.1.1 Problem Definitions

Suppose that a person with a large collection of images wishes to build an object recognition system to identify the images containing chairs, and to describe certain properties of the chairs that appear. One common way to think about this problem is to break it into a series of three sub-problems. The first step is to identify the images which contain chairs; the second is to examine each of those images to determine which portions of them contain the chair, and which portions contain other objects or the background. The third step consists of focusing on each image section containing a chair isolated from the background, and estimating the orientation, make and model, or other desired properties of the chair.

Most prior work in object recognition focuses on providing solutions to one of the three sub-problems. The first step, which we term *detection*, consists of identifying whether or not a particular image contains a particular target object, regardless of where the object

appears in the image or how much of the image it consumes. The second step, *localization*, is the problem of determining which portions of a particular image are taken up by target objects. For example, many localization algorithms identify rectangles of image pixels which circumscribe found target objects, while other algorithms report lists of pixels which project onto the target objects. Finally, for *property estimation* the problem is to determine physical properties of the target objects and the imaging process. Some example chair properties to estimate include pose [5][117], make and model, and configuration, *i.e.* whether the armrests are folded up or down. Other estimated properties include the expression (happy, sad...) and identity [128] of human faces.

As illustrated in the chair example, it is possible to think of the three sub-problems as successive modules in an overall system, instead of separate, isolated problems. However, it is not necessary to apply each of the three steps individually to every object recognition application; for example, it is common for human face detectors [104][118][3] to solve a localization problem by searching for faces in each portion of each image, without first detecting whether or not the image contains faces overall. For some applications, localization and property estimation may not be important. For instance, a common problem in medical image analysis is detecting which images contain target objects like tumors or other anatomical abnormalities [51][85], regardless of what parts of the images they are in.

This thesis is concerned with solving localization problems. Given an image, we wish to identify which sections of the image contains objects like chairs, ladders, and stools. In other words, given input images like those in the left column of Figure 1.1, we wish to obtain a result like the right column of Figure 1.1, in which the image has been annotated with which portion contains the object. We do not assume that the images have been passed through a detection phase, in other words we do not assume *a priori* that the image contains any target objects.

1.1.2 Sources Of Variability

The appearances of chairs can vary dramatically from chair to chair and image to image. The color, shape, and reflectance properties of a recliner will likely be very different from those of a metal folding chair, for example, causing images of the two types of chairs to differ significantly. Variability in appearance from image to image is the major obstacle in the development of object recognition systems– indeed, to date, no object recognition systems have been demonstrated which are able to detect and localize any and all of the world's chairs in arbitrary images. However, the designer of the chair-recognition system

may know that his collection of images is limited in scope– perhaps he knows that most or all of the chairs shown in the image collection are made of wood, or are in front of a blue screen, or appear in front of any other objects in the scene. In turn, the system designer can focus on accounting for high variability in some environmental factors like lighting, chair shape, or chair pose, while assuming that other factors– in this case, material properties, background clutter, or occlusion– are relatively constant.

Every object recognition algorithm makes simplifying assumptions about which factors related to the viewing environment, objects, and imaging process are likely to vary from image to image, and to what extent they are likely to vary. We refer to the factors that vary significantly as the *sources of variability* which the algorithm is designed to account for. In order to make incremental progress toward the long-term goal of recognizing a wide variety of objects under a wide variety of viewing conditions, it is common to develop algorithms which are able to accommodate certain sources of variability, and gradually generalize the algorithm so that fewer and fewer factors of the viewing process are assumed constant. For example, the single-pose face recognition algorithm in [8] is generalized to multiple poses in [43].

A widely-addressed source of variability is what we will call object *identity*. When we refer to object identity as a source of variability, we refer to the idea that the target "object" is in fact an abstract class that includes several physical entities which share some common characteristics. In the chair recognition example, if the image collection contains images of many instances of the same make and model of chair, or even images of completely different chairs, we would count object identity as one of the sources of variability for the algorithm to cope with. Referring to object identity as a source of variability is somewhat arbitrary; instead, it is possible to alternatively pose each instance or type of chair as its own distinct target object. For this discussion, however, we choose to think of a single target object, "the chair," whose appearance varies from image to image due in part to the fact that different instantiations of the chair are viewed. When object identity is a source of variability, authors often refer to object recognition as "category" or "class" recognition [34], since the notion of the target object generalizes to encompass multiple physical entities. Face recognition is another example application in which object identity is a source of variability; the target object is the set of all human faces, and the property estimation problem is to determine the identity [128].

In this thesis, the most important sources of variability are changes in the poses of objects, and changes in clutter. Specifically, we present algorithms for localization of objects

like chairs and ladders that are allowed to rotate arbitrarily in the ground plane and translate with respect to the camera between views. Objects in the background are allowed to change positions between images as well. We assume that camera parameters and lighting do not vary widely across all images. Also, we assume that the intrinsic properties like reflectance characteristics and geometric structure of our target objects do not change between views, and that their scale in the image does not vary significantly from view to view. In Chapters 3 through 6 we assume that our objects give rise to a reliable set of intensity edges in the image; in other words, edges in the image should correspond to occluding contours on our target objects rather than lighting effects like specularities or shadows. Finally, for the present work, identity is not a source of variability; that is, we seek to recognize specific physical instances of ladders, chairs, etc.

1.1.3 Cues

If our chair recognizer is to have any hope of success, the chairs in the image collection must be discriminable from other items in the images based on some set of characteristics of the images. For example, if all of the chairs in the image collection are known to have a pattern on the fabric– plaid, say– that distinguishes them from the rest of the environment, then searching for portions of images that look like the pattern is a reasonable strategy for detecting and localizing the chairs. On the other hand, if the images contain several different types of plaid objects, then plaid-like image patterns by themselves will not be sufficient to distinguish chair images from non-chair images. In this case, additional information extracted from the image– colors and shapes for example– may be useful to discriminate chairs from non-chairs.

All approaches to object recognition are predicated on an assumption that the target objects are discriminable from each other and from the viewing environment based on some set of image characteristics. We refer to these characteristics as visual *cues*. Researchers have exploited a wide variety of visual cues for recognizing objects, and difficulties related to cues remain central to object recognition research: these include efficiently representing cues like shape and texture in numerical entities called *features*, comparing cues from different images, and determining what the useful cues are for a particular recognition task.

We focus on exploiting two cues for recognition: shape and visual texture. By “shape” we are referring to the spatial configuration of image contours that represent the projections of objects. We emphasize that we are not referring to shape in terms of the intrinsic geometric structure of objects; instead, shape is an image property generated jointly by geometric

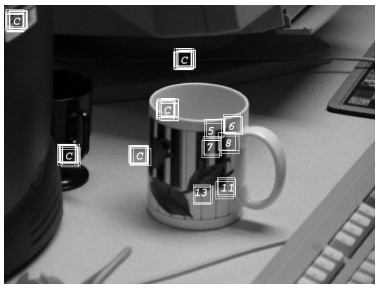


Figure 1.2: In Chapter 7, we address identifying parts of textured objects, like the mug. The rectangles represent numbered mug parts identified by our algorithm.

structure and imaging parameters. For example, in Figure 1.1, bottom row, shape cues are based on how the edges running along the legs, armrests, and other parts of the chair are arranged in the image. In Chapters 3 through 6, we examine the edge configurations to localize the target objects. The assumption is that the shapes that these edges form in chair images are distinctive from the shapes formed by edges drawn from images of other items. Other shape-based recognition methods of this variety include [9][72].

Visual texture, on the other hand, refers to distinctive image patterns, like plaid in the above example. In Chapter 7 we assume that visual texture is distinctive on sections of the objects we wish to localize, and we describe a procedure for detecting object parts based on those textures (Figure 1.2). Other texture-based approaches include [102][10].

1.1.4 Methods

Once the problem definition and sources of variability have been determined, and the designer has determined a set of visual cues which effectively discriminate chair images from non-chair images, strategies for representing how images of chairs tend to look, and for exploring new images for those appearances, must be determined. We will use the term *methods* to refer to the general computational strategies followed by recognition algorithms. There are many different ways to categorize algorithms according the methods they follow. Here, we present three criteria which we feel are useful in delineating the important methodological differences.

If the chairs are known to be plaid, it may make sense to search for chairs directly from the image signal, for example by decomposing the image into its frequency components and representing plaid images by distinctive characteristics (such as wavelet coefficients [96]) in the frequency domain. On the other hand, if the images of chairs are known to be well-represented as pairs of adjacent uniformly-colored blobs (the back and seat) connected to

four long, parallel lines below them (the legs), then a good approach may be to first search the images for uniformly-colored blobs and groupings of lines, and then examine how those blobs and lines are arranged [36]. Finally, if 3D models of the chairs are available, it may be possible to determine whether chairs are present in an image by determining whether there exists a 3D chair pose which would plausibly give rise to the image [46].

These three approaches illustrate what sorts of structures in the image or scene are detected as intermediaries on the way toward detection, localization, and property estimation. The first example is typical of *direct* methods which estimate no intermediary structures, instead representing images of objects directly from low-level signal properties; the second illustrates what we call *perceptual organization* techniques which estimate image structures like contours or coherent color regions; the third is a *physical estimation* approach which estimates properties of the scene such as 3D shapes and illuminants. In Chapters 3 through 6 we present a shape-based recognition technique which can either be applied directly to the image signal or can take advantage of estimated image contours. In Chapter 7, we present a direct method for localization based on efficient representation of the image signal in textured regions.

Another way to categorize previous approaches to recognition is by how they organize their search through the image for the target object. A *bottom-up* approach to chair localization might first scan the image in a search for image regions which look like legs, armrests, and seats, then aggregate the found parts into overall instances of chairs (as in, [19]); conversely a *top-down* approach would first scan the image at a coarse scale to identify large image regions which look as though they may contain chairs, then examine those regions in more detail for supporting evidence in the form of leg or armrest sub-regions [81]. Techniques which search for the target object in its entirety with no notion of component object parts we call *single-step* methods (for example, [82]); also, there are a few *hybrid* techniques which alternate between top-down and bottom-up phases [129], or provide a framework for fusing top-down and bottom-up algorithms together [126]. Top-down techniques can be useful if it is possible to quickly discard large image regions that are dissimilar to the target objects at a gross scale. Here, however, we focus on bottom-up methods in which a first step identifies individual image patches which project onto the object, and a second phase gathers these patches up into overall localizations of the object. The motivation for using bottom-up techniques is that for the complex-shaped objects we wish to recognize, it would be difficult for a top-down approach to efficiently discard object-sized sections of the image based on its gross shape characteristics. The main focus of this document is on the first phase of bottom-up recognition, namely identifying which pixels or patches be-

long to the target object. However, in Section 4.2 we also present a hybrid technique which alternates between pixel-level and object-level recognition processes.

A third major division among previous approaches to object recognition places *generative* techniques in one category and *discriminative* techniques in another. Cues extracted from images are summarized in numerical form by features; example features include the curvature of edges and the red, green, and blue components of color. A generative approach to chair recognition would formulate a statistical model which describes what sorts of values the features drawn from the image take on, given that a chair is present there. The parameters of this model can be estimated using example images of chairs [104] or CAD models [109]; performing recognition in the large collection of images consists of extracting the features and using the statistical model to infer the probability that chairs are in particular image regions, given the features. An analogous statistical model can be formulated which describes the probability of features taking on particular values in image portions containing the background or other objects. Objects are detected wherever the probability of chair is high, and the probability of background is low. Meanwhile, a discriminative approach to chair recognition partitions the space of all feature values into "chair" and "non-chair" categories. How to partition this space is determined by examining example images or 3D models of chairs and non-chairs, and in order to determine whether chairs are present in portions of novel images we extract features and determine whether they fall into the chair or non-chair category (see, *e.g.* [87]). For a more general discussion of the differences between generative and discriminative techniques, see [99].

This thesis is focused on discriminative methods for recognition. In particular, we focus on techniques for discriminating between target-object and non-target-object image sections, based on shape and texture features drawn from those sections. We determine the partitioning between the two categories of features by looking at features drawn from example images whose pixels have been labeled as belonging to the target objects or background. We focus on localization in high-clutter scenes; therefore, we feel that formulating recognition in terms of discriminating between object appearances and background appearances is a more direct way of addressing the problem than developing separate generative models for object and background classes.

To summarize, previous work in object recognition can be categorized according to four criteria. The problem definition and sources of variability describe the major characteristics of the problem to be solved, in terms of what questions to answer and what causes the problem to be difficult. The cues and methods describe the solution to the problem,

specifically the types of image information and computational methods used to arrive at a solution. We categorize the work in this document, along with some of the more significant prior approaches to recognition, in Table 1.1.

The focus of the work is on the estimation of image locations of objects across wide changes in pose and backgrounds; variability in object instance, articulation, lighting, and other properties are assumed small. We use bottom-up, discriminative techniques to localize the objects based on shape and visual texture. The algorithms developed here can encompass stand-alone applications which use training images to determine the presence and extent of objects in cluttered test images across wide changes in pose; or, as suggested above, our algorithms may also be useful as a pre-processing step for procedures which estimate physical properties like object pose.

1.2 Bottom-Up Localization

In this section we focus on bottom-up approaches to localization based on shape and texture cues. Previous approaches of this variety can be organized in terms of three constituent steps:

1. **Feature extraction**. As a first step, the image cues from each portion of the image are represented in numerical form. In Figure 1.1, for example, the image features could be numerical properties of the binary edges shown in the second column.

2. **Part classification**. Features from each portion of the image are classified as belonging to the target object or the background; in Figure 1.1, third column, white pixels represent sections of the image identified as a part of the ladder, cart, or chair. Additionally, this step may identify features with specific physical components of objects, like the seat or legs of the chair.

3. **Aggregation**. Once individual image sections corresponding to parts of target objects have been identified, these individual parts must be somehow joined together into overall hypotheses about how many objects are in the image, and where. Thus, this step would take the initial identifications of legs, armrests, and other chair parts and assemble them into a conclusion that a chair is present at a particular location in the image (Figure 1.1, fourth column).

Chapter 2 discusses related approaches to bottom-up localization in detail. Most of them can be well described in terms of local feature extraction, part classification, and aggregation, although in many approaches, one or more of the steps are quite simple. In this thesis, we present two types of bottom-up localization techniques: one focuses on image features, part classification, and aggregation based on shape cues, and the other is concerned with local extraction of texture features. Specifically, we present the following contributions:

1. **Cascade of Edge Probes – Chapter 3**. Here, local feature extraction consists of extracting binary edges, then representing local edge properties in terms of features which measure local edge density. We use a cascade of decision trees to classify these edge features as belonging to the target object or the background. Finally, features identified with the target object are aggregated together by scanning the image with an object-sized bounding box, and identifying overall instances of the object in locations where the number of object features is high. The thrust of this approach is on the representation and classification of the shape features.

2. **Cascade of Edge Operator Probes – Chapter 4**. Instead of image features based on binary edges, we extract a set of continuous edge features directly from the raw image. These features are classified using a cascade of decision trees, and parts are aggregated using bounding boxes, as for the cascade of edge probes. The problem of automatically selecting a set of edge features to use for image features makes this modification of the cascade of edge probes challenging. Again, the main focus here is on image features and part classification.

3. **Aggregation-Sensitive Part Classifier Training – Chapter 4**. Rather than processing the images in a strictly bottom-up manner, we alternate between classifying individual image parts and aggregating those parts into object instances using bounding boxes. Image features are based on shape, part classifiers are decision trees, and aggregation is based on scanning the image with a box; however, the part classification and aggregation alternate at run time.

4. **Discriminant filters – Chapter 7**. Discriminant filters extract texture features from patches in the image by projecting the patches into a low-dimensional space. The features are then classified as belonging to the target object or background using nearest-neighbor techniques or simple generative models. The focus of this approach is on

determining the projection from the image patches to the low-dimensional space, so we do not address aggregation here.

1.3 Applications

Applications of object recognition systems are spread across a wide range of areas including manufacturing, medicine, robotics, and surveillance. However, these applications can be placed roughly into categories according to the respective roles that the recognition systems and humans are expected to play.

In one set of applications, the object recognition system performs an estimation which is used as the basis for planning the actions of a robot or other automated system; in other words, the role of object recognition is to enable an automated task which does not involve humans at all. An example system that falls under this category is the Schneiderman face detector which has been incorporated into photograph developers in Europe[95]; once the recognition system detects faces in the photograph, another module automatically locates the eyes in the photograph and remove red-eye artifacts. Another prototypical recognition system of this type is the first phase of a "bin-picker": these systems estimate the poses of industrial gadgets on assembly lines so that robotic manipulators can grasp and move them[53]. A final example are "workhorse" robots; object recognition results based on images from cameras on the robot are used to plan interactions between the robot and the objects (for example, moving furniture from place to place) or to issue concise reports on the state of the environment for surveillance purposes. The algorithms developed in this thesis could be applicable in this context, for example to help robots determine the locations of chairs, ladders, and other objects they are intended to interact with.

Another common application area is "image mining." Here, the role of the object recognition system is to quickly summarize the contents of a large number of images and then present the summary to a human user for browsing. Common examples of applications in this area include searching through a large corpus of images and using the contents of the images to find those which contain objects with particular characteristics; this is commonly referred to as content-based image retrieval [116] and is exemplified by commercial products such as the QBIC system by IBM[119]. Another application in this area is the automated summary of surveillance video (see, for example, [23]); the role of the recognition system is to analyze hours of grainy video sequences and issue a concise report stating changes in the location or other state of objects seen in the filmed environment. Finally, an

emerging application in medicine is the use of recognition systems to both automatically detect objects such as tumors in large sets of biomedical images [32] and discover regularities among medical images which correspond to a particular medical condition[88]. The chair recognition example running through this chapter is an example of how our work can be applied to image mining applications; for example, our techniques can be used to detect chairs in images and help users to find all the images in a large collection which contain them.

A final set of object recognition applications involves neither a completely automated loop between the recognition system and another automated system, nor a single-step process of summarizing a large set of images for a human user, but instead an iterative loop of estimating object properties in images, presenting the results to an end user, and receiving feedback so that the recognition results are tuned to the changing needs of the user. The PicHunter system for image retrieval is an example system along these lines[20]. A retrieval system presents the user with a set of images sharing some common visual cues; the user provides the system with feedback as to which cues are relevant for his/her browsing purposes and the system adjusts its retrieval procedure accordingly. In this setting, we can imagine using our algorithms to present the user with initial guesses of where ladders, chairs, etc are found in images, and having the user click on the objects that interest him/her so that the parameters of our recognition system may be refined.

1.4 Conclusion

In summary, this thesis contributes discriminative, bottom-up and hybrid techniques which enable the localization of objects in cluttered scenes based on their shape and texture. We show bottom-up methods for shape-based localization in Chapters 3 and 4, as well as a hybrid shape-based algorithm in Chapter 4 and a bottom-up technique for texture-based localization in Chapter 7. The following chapters describe the algorithms, experiments which substantiate their usefulness, and an explanation of how these techniques are related to prior work.

Approach	problem definitions	sources of variation	cues	methods	references
Cascade Of Edge Probes	**localization**	**pose, clutter**	**shape**	**bottom-up, perceptual organization, discriminative**	**Chapter 3**
Cascade Of Edge Operator Probes	**localization**	**pose, clutter**	**shape**	**bottom-up, direct, discriminative**	**Chapter 4**
Aggregation-Sensitive Part Classification	**localization**	**pose, clutter**	**shape**	**hybrid, direct/ perceptual organization, discriminative**	**Chapter 4**
Discriminant Filters	**localization**	**pose, clutter**	**texture**	**bottom-up, direct, discriminative**	**Chapter 7**
Local PCA	localization	pose, occlusion, clutter	texture	bottom-up, direct, generative	[84] [56] [22]
PCA/Eigenfaces	detection, localization	pose, identity	texture	single-step, direct, generative	[82] [80]
Schneiderman	localization	identity, clutter, pose, lighting	texture	top-down, direct, generative	[104]
Viola-Jones	localization	identity, clutter, pose, lighting	texture	single-step, direct, discriminative	[118] [68]
Shape Indexing	properties	pose, occlusion	shape	bottom-up, perceptual organization, generative	[64] [6] [106]
Interpretation Trees	properties	pose	shape	top-down, perceptual organization, generative	[46]
Local Filter Banks	localization	pose, occlusion, clutter	texture	bottom-up, direct, generative	[10] [93]
Local Greylevel Invariants	detection, localization	pose, occlusion	texture	bottom-up, direct, generative	[45] [60] [71] [83] [100]
Receptive Field Histograms	detection	pose	texture	bottom-up, direct, generative	[102]
SEEMORE	detection	pose, deformation	texture, shape, color	bottom-up, direct/perceptual organization, discriminative	[78]
Probabilistic Contour Discriminants	localization	deformation, occlusion, clutter	shape, texture	top-down, perceptual organization, generative	[73] [74]
Data Driven MCMC	localization	occlusion, clutter	shape	hybrid, perceptual organization, generative	[129]
Fisherfaces	properties	identity, lighting	texture	single-step, direct, discriminative	[7]
Mohan	localization	identity, pose, clutter	texture	top-down, direct, discriminative	[81]
3D Shape Signatures	properties	pose, clutter	shape	bottom-up, physical, generative	[55]
Convolutional Neural Networks	properties	deformation, clutter, identity	shape	single-step, direct, discriminative	[66]
Illumination Cones	properties	lighting, identity, pose	texture	single-step, direct, generative	[43] [8]
Li and Hornegger	localization	pose, clutter, occlusion	shape	bottom-up, perceptual organization, generative	[67]
Local Color Invariants	localization	lighting, pose, clutter, occlusion	color	bottom-up, direct, generative	[110]
Boykov and Huttenlocher	localization	pose, clutter, occlusion	shape	bottom-up, perceptual organization, generative	[13]
Papageorgiou *Et Al*	localization	clutter, identity	texture	single-step, direct, discriminative	[87]
Boosted Discriminants	localization	clutter, identity, pose	texture	top-down, direct, discriminative	[76]
Shape Contexts	detection	identity, deformation, pose	shape	bottom-up, perceptual organization, generative	[9]
Amit, Geman, Jedynak	localization	identity, deformation	texture, configuration	bottom-up, direct, discriminative	[3]
Body Plans	localization	identity, deformation, clutter	texture, color, configuration	bottom-up, perceptual organization, generative	[36]
Similarity Templates	localization	identity, clutter	color	single-step, direct, generative	[111]
SIFT	properties	clutter, pose, occlusion	texture	bottom-up, direct, generative	[71]
Primitive Grouping	localization	identity, clutter, occlusion, pose	shape	bottom-up, perceptual organization, generative	[72] [50] [37]

Table 1.1: Examples of prior approaches to object recognition, categorized according to the taxonomy in Section 1.1. The techniques presented in this thesis are highlighted in the first four rows.

Chapter 2

Related Work

As described in the introduction, this thesis is focused on methods for bottom-up localization based on shape and texture cues. In this chapter we explain how our techniques fit into the context of prior bottom-up approaches in terms of their constituent steps: feature extraction, part classification, and aggregation. To facilitate this discussion, we begin by stating each step in a more precise mathematical form:

- **Feature extraction** Feature extraction is the process of mapping image patches or local groupings of line segments to a low-dimensional space. For many approaches, feature extraction begins with extraction of a set of rectangular patches $\{P\}$ from the image, for example by scanning the image with a rectangular box. A function f then maps each P to a low-dimensional vector $\mathbf{v} = f(P)$. The vectors $\mathbf{v}$ are what we refer to as the *image features*. In many texture-based approaches, $f(P)$ consists of raster-scanning P into a vector $\mathbf{p}$, and applying a linear transformation T to $\mathbf{p}$. In many edge-based techniques however, binary edges are first extracted from the image, and the edges are clustered into local groupings, for example by searching for groups of edges that are connected, parallel, or lie within a small image patch. A function f then maps each local edge grouping to a low-dimensional space. For example, given edges $\mathcal{L}$, $\mathbf{v} = f(\mathcal{L})$ may contain entries summarizing the orientations or curvatures of the edges in $\mathcal{L}$.

- **Part Classification** Feature extraction results in a set of vectors $\{(\mathbf{v}, \mathbf{b})\}$ such that $\mathbf{b}$ summarizes the image location of the patch or edge grouping corresponding to $\mathbf{v}$ (for example, $\mathbf{b}$ could contain the coordinates of a bounding box around a patch). The purpose of part classification is to label each of the vectors $\mathbf{v}$, resulting in a

new set $\{(\mathbf{b}, l, w)\}$. Each label l corresponds to either a part of a target object or to the background, and optionally w represents the confidence or score given to that labeling. In some cases, the labels signify a specific physical part of the target object, like the mouth or nose of a human face [19]; in other cases the labels are less specific, signifying only that $\mathbf{v}$ corresponds to some (*i.e.* any) section of a target object, as opposed to the background [56].

- **Aggregation** The aggregation step uses the set $(\mathbf{b}, l, w)$ of image locations, part labels, and scores to produce a set of object instances $\{(o, \boldsymbol{\theta})\}$, where each o is a label indicating which target object is present, and $\boldsymbol{\theta}$ is a vector of parameters indicating the location of object o in the image. $\boldsymbol{\theta}$ may consist of a list of all pixels which project to the instance of o, the coordinates of a bounding box that circumscribes o, or some other representation of overall object location. The aggregation step can generate object instances by reasoning about how the parts are arranged spatially in the image for instance.

Thus, the three chief characteristics that determine a bottom-up approach to localization are the feature mapping f from image patches or edge groupings to a low-dimensional vector space; the classification scheme for applying labels and scores to image features; and the aggregation scheme for generating hypothesized object instances from the labeled features. This document presents algorithms for these three steps based on edge and texture information. In Chapters 3 through 6, features measure edge properties in local image neighborhoods, part classification is accomplished using a series of decision trees, and aggregation consists of collecting classified parts into rectangular image boxes. In Chapter 7, on the other hand, image features are discriminative measures of local texture, and the choices of part classifiers and aggregation steps are left open. The rest of this chapter describes how the techniques in this thesis relate to prior work in object recognition in terms of these three steps.

Central to our analysis is the role of *training data* in designing the feature extraction, part classification, and aggregation steps. That is, consider a set $\mathcal{P}_t = \{(P_t, t)\}$ of *training patches* P_t drawn from example images, together with labels t which identify P_t with an object part or background. P_t could be used to drive the design of the feature mapping f. For example, f could be tuned to map example patches corresponding to different parts to distinct features, in other words forcing $d(f(P_a), f(P_b))$ to be large for P_a and P_b corresponding to distinct parts, under some distance metric d. Once a feature mapping has been determined, the set $\{(\mathbf{v}_t, t)\}$ can be employed in the design of the part classifier,

namely by encouraging classifiers which label each $f(P_t)$ with the correct t.

In some cases, training patches corresponding to parts of the target object are employed to optimize one or more of the bottom-up steps, but training patches corresponding to the background, *i.e.* the viewing environment or clutter objects, are not. Example images containing clutter may be unavailable; for instance, images of target objects in isolation or 3D models of isolated target objects may be the only source of data at hand during the training process. However, our main point in this chapter is that when example images containing both object and background are available at training time, both types of example patches– those containing parts of target objects and those containing clutter– should drive the design of image features, part classifiers, and aggregation steps. In short, each step should focus on discrimination between features, parts, and sets of parts belonging to objects and clutter.

2.1 Feature Extraction

The task of determining the mapping f is often referred to as *feature selection*. In Chapter 3 we present a shape-based recognition technique in which we automatically determine a set of shape features which discriminates between groupings of binary edges corresponding to target objects and groupings of binary edges corresponding to background. Chapter 4 extends this approach to more expressive shape features extracted from the raw image. Additionally, in Chapter 7 we present a technique for selecting features such that texture patches from object and background are well discriminated. In this section we categorize previous bottom-up approaches to shape-based and texture-based localization according to their feature selection strategies. Our overall critique of prior work is that in most cases, the criteria for determining good feature mappings are motivated by concerns other than discrimination between parts of the target object and parts of the background. As a result, prior techniques can be prone to ambiguities between object features and background features.

For some texture-based approaches, the chief criterion for choosing the feature mapping is good approximation of patches; that is, given a set $\{P_t\}$ of example image patches of parts of our target objects and background, the goal is to find a mapping f such that the reconstruction error $d(P_t, f^{-1}(f(P_t)))$ is minimized according to a distance metric d. Several authors, including [22], [84], [104], and [56] propose the use of principal components analysis (PCA) in this setting, so that f consists of a linear transformation of P_t, and d is

the L_2 norm. PCA can be a good way to approximate the appearances of image patches, however for our application we are more interested in discriminating between sets of image patches than reconstructing them. In particular, local PCA does not necessarily encourage partitions between features corresponding to target objects and features corresponding to the background. Specifically, suppose our set of example patches consists of two subsets, $\{P_{t+}\}$ and $\{P_{t-}\}$, where each P_{t+} is an image patch that projects onto a target object, and each P_{t-} maps onto the background. PCA features $f(P_{t+})$ and $f(P_{t-})$ may be accurate low-dimensional approximations to P_{t+} and P_{t-}, but PCA makes no attempt to ensure that $d(f(P_{t+}), f(P_{t-}),)$ is high, in other words that $f(P_{t+})$ and $f(P_{t-})$ are distinguishable from each other. As a result, it is necessary to tune parameters– namely, the image patch size and number of eigenvectors in the PCA projection– to ensure that features corresponding to the object and background are not confused.

Instead of extracting features which approximate image patches well, other techniques focus on extracting features which are invariant to class of transformations of the patches. In other words, given a class $\mathcal{T}$ of image transformations such as rotations, scalings or skews, invariant features are constructed such that for any image patch P, and any transformation $T \in \mathcal{T}$, $f(P) = f(T(P))$. For example, texture features have been presented which are invariant to affine transformations [60][45][83][100][10], scalings [34], and rigid body transformations [103]. Additionally, local affine-invariant edge-based features are widespread in techniques based on matching local shape features, for example [64][67][106][121][71]. Using invariant features can simplify the design of part classifiers since the feature values are guaranteed to stay constant despite certain changes in viewpoint. However, there is still no guarantee that invariant features extracted from portions of the image corresponding to the target objects, and invariant features drawn from background portions of the image, will take on distinct values.

A third style of feature selection is to choose a feature mapping f so that the resulting features are expressive and capture salient aspects of image regions, regardless of whether the features reconstruct the image well or are invariant to transformations. For example, shape contexts [9] are local edge-based features which measure the density of edge pixels in image neighborhoods surrounding a pixel of interest. They are not strictly invariant to any image transformations other than translations, and they are not designed to reconstruct the edges in the image well, but they still capture salient local shape properties in terms of how edges are distributed spatially. Similar arguments for rich shape descriptors which lack invariance properties may be found in [6]. Responses to sets of Gabor filters [81][102] or derivative-of-Gaussian filters [102][93] have been utilized as texture features, partially

based on biological motivations, and partially based on the desirable signal properties of the filters. Again, these features are not designed with discrimination as a design criterion, and therefore their design parameters must be tuned to ensure that features drawn from images of target objects are discriminable from features drawn from images of the background.

Finally, image features can be selected so that they optimize a discrimination criterion; in other words, it is possible to choose f so that features corresponding to patches or edges of distinct object parts or background are distinct from each other. Specifically, consider a set $\{(\mathcal{L}_t, t)\}$ of local edge groupings $\mathcal{L}_t$ drawn from example images, together with their part labels t. A discriminative approach to feature selection chooses f so that given a distance metric d, $d(f(\mathcal{L}_a), f(\mathcal{L}_b))$ is high for distinct parts a and b. The motivation for optimizing discriminative criteria during feature selection is to ensure that the feature mapping does not essentially collapse image patches or edge groupings from distinct object parts into identical or highly similar features.

In fact, feature mappings which optimize discrimination criteria are not common in bottom-up object recognition. However, discrimination does drive feature selection in some single-step and top-down texture-based methods. For example, convolutional neural networks [66][65] optimize the coefficients of convolution and downsampling operations so that the outputs of these operations across different classes of images (for example, images of different digits) are discriminated. Similar neural network architectures for single-step localization [98] can be thought of as training the system to optimize the extraction of texture features for discrimination purposes. Some single-step techniques use training procedures to find a linear discriminant which discriminates images of different target objects from each other [7][87]. Other single-step detection methods use discriminative criteria to drive the search for a set of vectors on which to project images for recognition purposes; these vectors can be found one at a time [118][68][58] or all at once [70].

Discriminative selection of shape features is also not common. Martin *et al* [77] have addressed the general problem of discriminating between edges corresponding to object boundaries and other types of edges, but not in a recognition context.

Outside of object recognition, discriminative feature extraction has been described in papers on texture discrimination [92][122] and image retrieval [114]. For example, Randen and Husøy[92] optimize the coefficients of global linear filters so that maximizing the separation of their responses to two textures becomes an eigenvalue problem similar to that found in the formulation for discriminant filters in Chapter 7. However, some assumptions about their texture discrimination scenario restrict the applicability of their approach to

bottom-up localization. For example, Randen and Husøy assume that the texture patches are separable autoregressive fields, while Weldon and Higgins[122] consider patches that are well-modeled as a dominant sinusoid plus bandpass noise. In image retrieval, Vasconcelos [114] selects local texture features which maximize a particular discrimination criterion between different categories of images. However, to our knowledge, selecting feature mappings which optimize discrimination criteria has not been presented in bottom-up localization.

2.2 Part Classification

Part classifiers typically use a set of image features drawn from example images or 3D models to train a part classifier which labels features drawn from novel images. These classifiers can be categorized as either *generative* or *discriminative*. The generative approaches use the examples to estimate how image features corresponding to parts of target objects and clutter are distributed in the feature space; that is, generative approaches attempt to approximate the density function $p(\mathbf{v}, l)$ which gives the probability that a particular image feature $\mathbf{v}$ co-occurs with part label l. Discriminative approaches partition the feature space into sections corresponding to distinct parts and background. In other words, a function $p(\mathbf{v})$ is derived which returns the label corresponding to the partition that $\mathbf{v}$ lies in, and optionally a confidence w in that categorization. Confronted with an image feature $(\mathbf{v}, \mathbf{b})$ drawn from a novel image, generative approaches return a set $\{(\mathbf{b}, p(\mathbf{v}, l), w)\}$ such that $p(\mathbf{v}, l)$ is high, or such that likelihood ratios $p(\mathbf{v}, l)/p(\mathbf{v}, a)$ are high for all $a \neq l$. Discriminative approaches return $\{(\mathbf{b}, p(\mathbf{v}), w)\}$.

In this document, we advocate discriminative approaches for part classification; in Chapter 3 we use recursive axis-parallel partitions of the feature space to categorize shape-based image features as belonging to target objects or clutter. We prefer discriminative techniques because they make fewer assumptions about how the feature values are distributed and generally lead to an easier estimation problem than generative approaches do. In particular, generative techniques require the estimation of $p(\mathbf{v}, l)$ for all possible $\mathbf{v}$ and l. The space of all $(\mathbf{v}, l)$ can be large, especially if $\mathbf{v}$ is high-dimensional, so in practice generative approaches assume that the feature values for each class are distributed according to a specific functional form, and then estimate the parameters of that function. Discriminative methods, on the other hand, only need to estimate a set of partitions to split the space of all possible $\mathbf{v}$ into regions corresponding to a particular l. In practice, the number of par-

titions needed to effectively split the space of **v** into coherent regions is often small. As a consequence, estimating partitions of the feature space is often easier than determining a functional form for $p(\mathbf{v}, l)$ and estimating its parameters.

Most prior approaches to part classification in bottom-up localization are generative in nature. In particular, many of them represent the density function p using the training examples themselves, and employ k-nearest-neighbor techniques to evaluate $p(\mathbf{v}, l)$ for a novel image feature **v**. That is, the part classifier searches for a set of k training examples $\{(\mathbf{v}_l, l)\}$ for each label l such that $d(\mathbf{v}, \mathbf{v}_l)$ is low according to a metric d, and then combines these distances in some way to arrive at $p(\mathbf{v}, l)$. "Voting" schemes can be thought of setting $p(\mathbf{v}, l)$ to a constant c for all labels l for which $d(\mathbf{v}, \mathbf{v}_l)$ is low. Some of the many part classification approaches which assign labels based on nearest neighbor criteria include [10] [21] [103] [93] [106] [56] [84] [9] [64]. This technique is simple and widespread, and it can often give good classification results in spite of its simplicity. However, the behavior of these techniques is determined by the distance metric d; one that is poorly designed may incorrectly assign high scores to labels whose training features are not highly similar to image features seen at run time. Most previous researchers in object recognition set the parameters of the distance metric arbitrarily, leaving them open to ambiguities between parts. One exception is Mahamud [75], who recently proposed using training examples to construct a distance measure d by linearly combining elementary distance measures $\{d_1, d_2 \cdots\}$ in such a way that ambiguity between features is reduced.

More elaborate parameterized models of $p(\mathbf{v}, l)$ have also been explored in the literature. For example, some researchers in bottom-up recognition [22] have suggested estimating an interpolating surface between the various training features $\mathbf{v}_l$ corresponding to a particular part l, and measuring the distance between **v** and this surface as an estimate for $p(\mathbf{v}, l)$. Rikert *et al* [96] fit a mixture-of-Gaussian distribution to the image features. Generally speaking it is difficult to estimate these models, and with interpolating surfaces there is still the problem of how to determine a distance metric between new image features and the model.

Discriminative part classifiers in bottom-up recognition are not as common. However, they do appear in some top-down approaches. For example, Mohan [81] employs support vector machines to discriminate between image features corresponding to distinct parts in his top-down technique.

2.3 Aggregation

Given a set of labeled image features, the aggregation step assembles the indicated parts into overall instances of target objects. Previous approaches to aggregation may be categorized by how they reason about the spatial configurations of parts in the image in order to put them together into instances of objects. In our shape-based localization techniques (Chapters 3 to 6), we focus on aggregation tools which add up quantities of interest at pixels inside a bounding box, without considering how those pixels are distributed spatially. As we explain in Section 3.5.5, we employ this simple approach to aggregation because it is much faster to apply to images than techniques that reason about the 2D or 3D pose of the target object. Furthermore, we wish to explore recognition scenarios in which detailed knowledge of the 2D or 3D pose of the target object is not important for the application.

The simplest approach to aggregation is to not consider the spatial arrangements of parts at all; especially in the case of detection tasks, it is common to simply add the scores of all labeled image features $(\mathbf{b}, l, w)$ corresponding to parts l drawn from the same object, and declare that the object is present somewhere in the image if the sum of scores, *i.e.* the *aggregate score*, passes a threshold [56] [64][10][21][103][93][106][56][84]. It is difficult to set this threshold properly so that true instances of objects produces aggregate scores higher than the threshold and aggregate scores for false instances of objects fall below it. Also, for simple shape features it has been shown that image noise can cause aggregate scores for false instances to be virtually indistinguishable from those for true instances [47]. We note that ignoring the spatial layout of identified object parts can lead to serious errors during aggregation. For example, individual identifications of two eyes, a nose, and a mouth will lead to a match for a face, even if the mouth is between the eyes and nose, the right eye is to the left of the left eye, and so on. Falsely identified parts scattered randomly in the background will generate false instances, even if those parts are not plausible in concert.

Schmid *et al* [103] propose two heuristics to incorporate the geometric configurations of parts into aggregation. First, labeled parts are discarded when they are nearby many labeled parts belonging to other objects. Also, pairs of parts are discarded when image gradients at their image locations are inconsistent with gradients in training patches for those parts. The first heuristic does not help when false matches for many parts of the same object are found in one area of the image. Also, the second heuristic is only useful when false part matches occur at random orientations in the image plane, and when the gradient is computed reliably.

In contrast, Perona *et al* [19] [34] construct a probabilistic model of where parts should be seen in relation to each other. The image positions of the set of parts corresponding to a particular object are assumed to be distributed according to a multivariate Gaussian distribution whose mean and covariance are estimated from training data. At run time, the "shape likelihoods" of sets of labeled parts are determined by computing their joint probability with respect to the distribution, and sets of parts with high probability are indicated as instances of the object. While this approach has been successfully applied to bottom-up recognition of some objects, the main limitation is that it is intractable to accurately estimate the distribution of part configurations when the number of parts is high. Also, since this technique is not discriminative, there is no guarantee that the configurations of falsely-identified parts among the background will be discriminable from the arrangements of true parts.

Other authors have proposed different probabilistic models to constrain the spatial layout of parts in the image. For example, several authors propose Markov models in which the probability that a particular set of object parts is present at a particular location in the image depends on how the parts are distributed over a Markov random field [67] [13] or Markov chain [74] overlaid onto the image. Aggregation proceeds by inferring, via the labeled object parts, high-probability arrangements of parts in the Markov model. Each of these methods restrict how parts can be arranged in instances of objects, either forcing them to lie on a parameterized family of curves [74] or constraining them to Euclidean [13] or similarity [67] transformations of a rigid configuration of parts. Furthermore, [13] and [67] are limited by computational concerns since they try to find the maximum *a posteriori* part positions and configurations over the entire image at once; the optimization problem in [67] in particular grows in complexity with the number of false part identifications.

Most top-down methods also restrict how parts are distributed in the image [104][81]. Here, a sliding window is passed over the image, and restricted sub-windows at each location are searched for the parts. Constraining part locations to sub-windows speeds aggregation, and the reduced complexity helps to offset the cost of evaluating every image location as a possible object location; however, it also means that objects can only be recognized over a limited range of aspects. Typically, this limitation is overcome by designing different recognizers for different poses; for example, we could slide one window over the image to search for frontal faces, one for profile-view faces, etc. In our work, however, we wish to formulate a unified algorithm to identify objects over all possible aspects in one pass.

In Chapter 3 we focus on feature extraction and part classification, and employ a simple

aggregation step which scans the image with a bounding box, searching for regions of the image with a large number of labeled object parts. This aggregation technique is clearly sub-optimal but it is simple to implement and allows us to test our image features and part classification techniques in the setting of an end-to-end system. Moreover, it begins to address an area that is lacking in the literature: the bottom up localization– not detection or property estimation– of objects based on edges.

Chapter 3

Shape-Based Localization

In this chapter we address the problem of using shape cues to localize wiry objects under significant variability in pose and clutter. More specifically, we address the localization of objects such as the chair in Figure 3.1, which appears at an arbitrary orientation with respect to the camera, and arbitrary translation along the ground in the direction parallel to the viewing plane. Given an input image (Figure 3.1, first column), we extract binary edges (Figure 3.1, second column) and use image features based on the local configurations of the edges to determine which edge pixels belong to an instance of a target object, and which edge pixels belong to clutter (Figure 3.1,third column). Once the edge pixels have been classified, we aggregate the edge pixels on target objects into overall hypotheses about the presence of the object by scanning the image with a box-shaped filter (Figure 3.1, fourth column). Alternatively, the classified edge pixels may be provided as input for property estimation procedures such as pose estimation[5][117]. Section 3.5.5 describes the aggregation scheme; however, the bulk of this chapter is concerned with the preceding problems of feature extraction and part classification in edge images. Specifically, we demonstrate a shape-based algorithm which localizes complex objects appearing at an arbitrary out-of-image-plane rotation in cluttered scenes. We show that our bottom-up approach enables us to localize common, complex-shaped objects at a variety of poses among highly cluttered backgrounds.

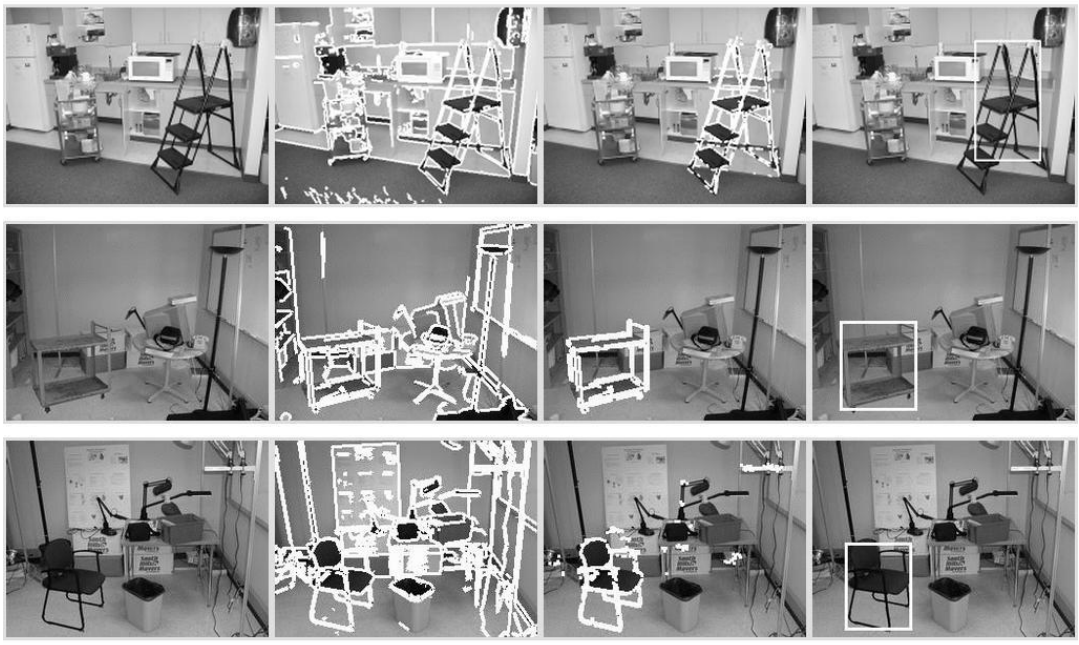


Figure 3.1: This chapter is concerned with edge-based approaches to bottom-up localization of wiry objects. Three example results are shown here. Left to right: input images, detected edges, part classification results, and aggregation.

3.1 Motivation : Wiry Objects

Over the past 10 years, significant progress has been made toward the recognition of real, complex objects in cluttered scenes. There are now object recognition systems whose detection and false alarm rates are encouraging for real-world applications; recently real-time detectors with reasonable performance have even emerged[118][105]. The most common target object searched for is the human face, but in principle these systems could be trained to detect any of a variety of objects including cars and buildings.

Many of the approaches use single-step methods. In particular, they model the appearance of a rectangular image patch circumscribing the object, across changes in pose [82], lighting [8], or other conditions. This reduces the recognition problem to examining a specific rectangular image template and using the appearance of the template to decide whether or not it surrounds an instance of the target object.

Since the problem is formulated in terms of rectangular image windows, single-step appearance-based methods work well when applied to target objects whose projection into the image fills a rectangular region. However, many objects produce images poorly approximated by rectangles; for objects such as the ladder, cart, and chair in Figure 3.2, their bounding boxes in the image will contain a high percentage of pixels which map to the

Figure 3.2: Examples of wiry objects.

background or other objects. Most successful recognition techniques can handle the variation in template appearance induced by a small number of background pixels in the image patch. When most of the template consists of clutter, however, its appearance can vary widely due to a modification of the background or object pose, making it difficult to detect the object based on the entire template. This chapter focuses on the recognition of objects like those in Figure 3.2; we will refer to them as *wiry* objects since they consist mainly of elongated, thin stick-like components connected together to form complex structures.

As an alternative, several authors [10][22][63][106] have proposed bottom-up methods based on the appearances of small patches covering parts of the object. This strategy may be effective for some objects; consider, however, wiry objects such as the chair in Figure 3.2. Any patch larger than a few pixels across will intersect mainly clutter pixels when placed over the legs or armrests, and it is doubtful that image patches a few pixels wide will contain sufficient information to discriminate the appearance of the object from the background.

Furthermore, many popular approaches to object recognition analyze the visual texture patterns in candidate patches; thus they work well when the target object has significant visual texture. Faces, cars, and buildings often possess this characteristic. But for the near-monochrome objects in Figure 3.2, along with many other common objects, there is too little appearance variation across the object surface to use texture as a cue for discrimination. Thus, while patch-based techniques are effective for some objects, we feel it is

worth investigating the problem of bottom-up localization from alternative cues, especially shape. Specifically, we employ discriminative machine learning techniques to boost the effectiveness of the shape-based recognition paradigm popular in the 1980s [46] to the point of feasibility in high-clutter scenes under significant 3D pose variation.

3.2 Problem Statement

We assume that target objects are characterized by their shape properties, and that these shape properties are reliably captured in the form of binary edges extracted from the object as it appears in images. Therefore, our bottom-up approach to localization focuses on identifying edges in the image corresponding to the target object. Our assumption is that identifying the object edges will be sufficient for identifying the object as a whole. Specifically, our bottom-up approach identifies individual edge pixels corresponding to target objects, and assembles the identified pixels into overall instances of the objects.

More formally, let $\mathcal{G}$ denote a list of pixels $\mathbf{q} = [x, y]$ from a novel image I such that an edge has been detected at $I[\mathbf{q}]$. Our goal is to use $\mathcal{G}$ to recover a second list, $\mathcal{O}$, which contains only those edge pixels $\mathbf{q} \in \mathcal{G}$ which correspond to points on an instance of a particular target object of interest. In more detail, our objective function for this task is to maximize tp_p, the number of *true positive pixels* in $\mathcal{O}$; and minimize fp_p, the number of *false positive pixels* in $\mathcal{O}$. True positive pixels are edge pixels $\mathbf{q} \in \mathcal{O}$ that project onto the target object, and false positive pixels are $\mathbf{q} \in \mathcal{O}$ that project onto the clutter. Object pixels we accidentally fail to include in $\mathcal{O}$ we refer to as *false negative pixels* and background pixels we (correctly) exclude from $\mathcal{O}$ are *true negative pixels*. False negative pixels and false positive pixels are mistakes: they are the object pixels we accidentally classify as background and the background pixels we accidentally classify as object, respectively. In Section 3.5.4, we discuss the fact that the relative importance of the two kinds of mistakes may vary from problem to problem, and we provide classification techniques based on analysis of *receiver operating characteristics* to address this issue.

We reach our overall goal of localizing instances of the target objects using the object edge pixels in $\mathcal{O}$. Earlier approaches to bottom-up recognition from edge cues aligned edges with detailed 2D or 3D object models in order to estimate pose or deformation of objects (see, for example, [9][46]). Here, however, we assume that we have no access to 2D or 3D object models in advance, and therefore must use clustering techniques to group the individual edge pixels into image regions corresponding to overall object instances. We

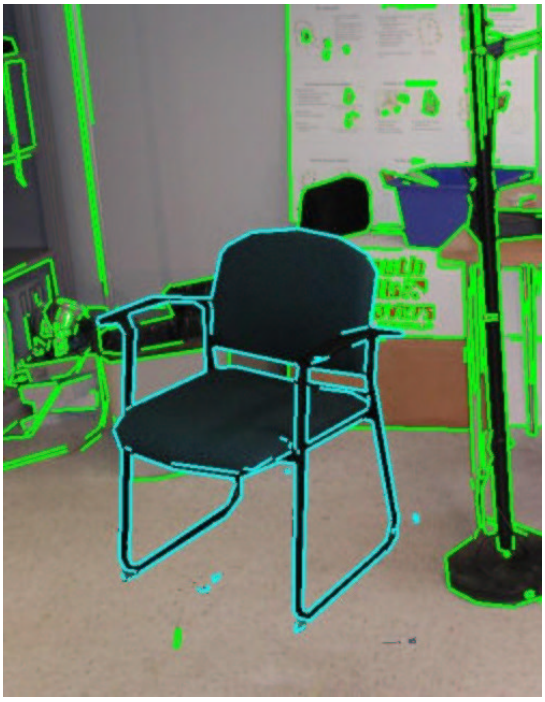


Figure 3.3: Example training image for the cascade of edge probes. Edges on the target object are marked in blue, and edges on the clutter are marked in green.

focus on localizing the object to rectangular image regions for speed and simplicity.

More formally, we aim to summarize the edge pixels in $\mathcal{O}$ into a list of object instances $\{(o, \boldsymbol{\theta})\}$, where each $\boldsymbol{\theta} = [x_{begin}, x_{end}, y_{begin}, y_{end}]$ represents the top left and lower right coordinates of a bounding box around an instance of the target object. Our only source of training data is a set of images containing the target object in typical scenes, from which edges have been extracted and labeled "object" or "clutter." In other words, at training time we are given a set of images $\{T\}$. Binary edges are detected in each training image and the edge pixels are partitioned into two lists, $\mathcal{T}_{T+}$ and $\mathcal{T}_{T-}$, containing the object and background edge pixels from T. Figure 3.3 shows an example training image.

As described in Chapter 2, a bottom-up solution to this problem proceeds by first breaking up the edges in $\mathcal{G}$ into a set $\{\mathcal{L}\}$ of local edge groupings, then classifying the edge groupings to object parts based on features extracted from them, finally clustering the classified edge groupings together into overall object instances. Since our primary goal is to determine which edge pixels are in $\mathcal{O}$, we make one grouping $\mathcal{L}_{\mathbf{q}}$ per edge pixel $\mathbf{q} \in \mathcal{G}$ and use the spatial configuration of edges in $\mathcal{L}$ to classify $\mathbf{q}$ as object or background. Specifically, $\mathcal{L}_{\mathbf{q}}$ contains $\mathbf{q}$ and all other edge pixels within a radius r from $\mathbf{q}$. In the context of classifying $\mathcal{L}_{\mathbf{q}}$, we refer to $\mathbf{q}$ as the *query edge pixel*, and the circle of the image containing

$\mathcal{L}_{\mathbf{q}}$ the *aperture* around **q**. Figure 3.6(a) shows an example image in which a query edge pixel is marked with an "x", and the black circle represents its aperture. Determining the aperture size is an important problem in this framework; in Section 3.5.2 we provide a solution in which we start with a small aperture and increase its size until **q** is identifiable as object or background.

The training data does not specify which distinct parts of the target object the edge pixels belong to; in the example of localizing chairs, this means that the set $\mathcal{T}_{T+}$ only lists which edge pixels project to some (*i.e.* any) portion of the chair, without giving details about which of them project to the legs, the seat, and so on. Thus, part classification consists of a binary task of determining whether a particular local edge grouping is to be identified as an anonymous portion of the target object, or as clutter. In other words, part classification maps the set of image features $\{(\mathbf{v}, \mathbf{b})\}$ to triples $\{(\mathbf{b}, l_+, w)\}$, where l_+ refers to the target object and w measures our confidence that the edges at **b** correspond to l_+. Since each vector of image features **v** corresponds to unique query edge pixel **q**, we will later refer to a *score image* W, where $W[\mathbf{q}] = w$ if the part classifier returns a triple $(\mathbf{v}, l_+, w)$, $f(\mathcal{L}_{\mathbf{q}}) = \mathbf{v}$, and $W[\mathbf{q}] = 0$ otherwise.

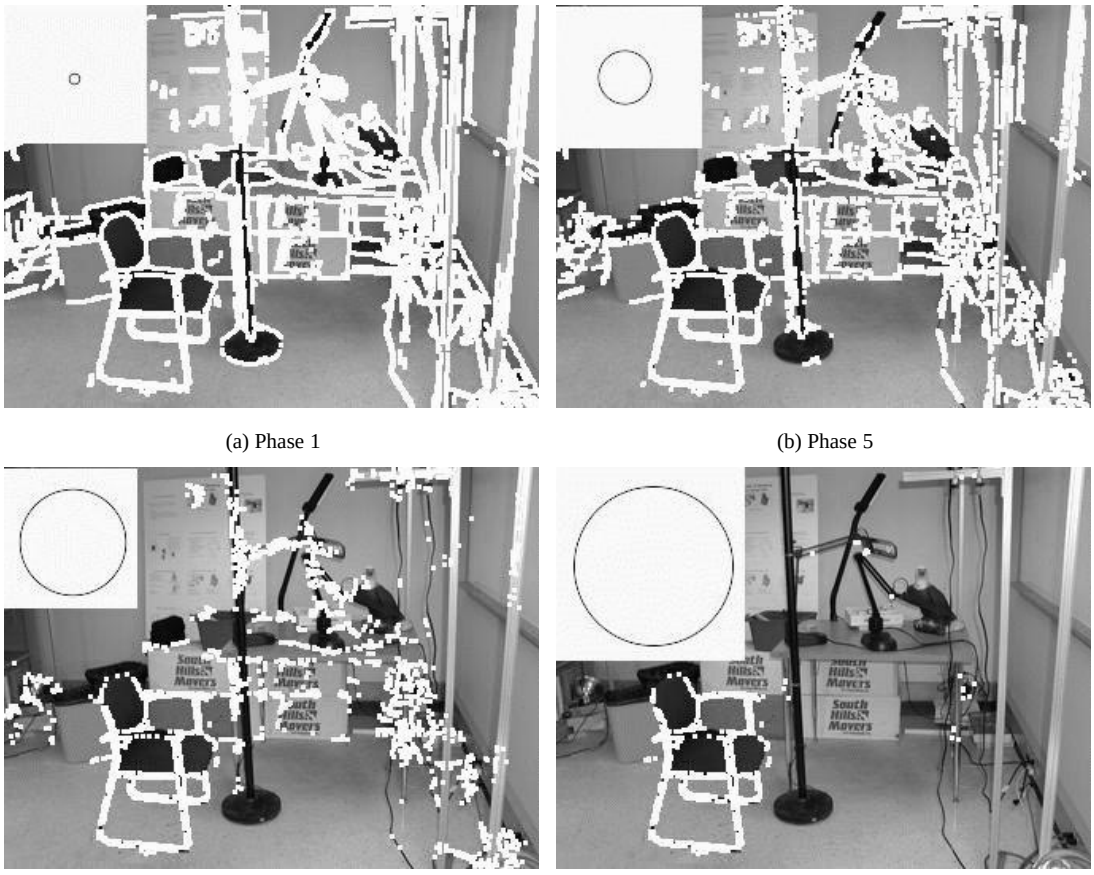
3.3 Solution Overview

Here is a brief sketch of our algorithm in terms of the bottom-up localization steps:

- **Feature Extraction** The image features extracted from a local edge grouping $\mathcal{L}_{\mathbf{q}}$, called *edge probes*, are measurements of the local edge density at fixed spatial offsets with respect to **q**. The spatial offsets are spread over the aperture; they are represented as "+" in Figure 3.6(a) for the **q** located at the "x."

- **Part Classification** We employ a cascade of decision trees to classify the edge probes. Successive classifiers compute edge density features over successively larger areas of the image to identify query edge pixels as object or clutter. Figure 3.4 shows an example of processing an image through several cascade phases, and Figure 3.5 depicts how each edge pixel is classified by decision trees in the cascade.

- **Aggregation** Once all edge pixels have been classified as object or clutter, we scan the image with an *aggregation filter*, essentially a rectangle that searches for portions of the image containing a high concentration of object edge pixels. If the number of



(a) Phase 1 (b) Phase 5

(c) Phase 10 (d) Phase 15

Figure 3.4: Example recognition results at successive phases of the recognition cascade. The size of the aperture for each phase is depicted by the circle at upper left. Edge points classified as “chair” are shown in white.

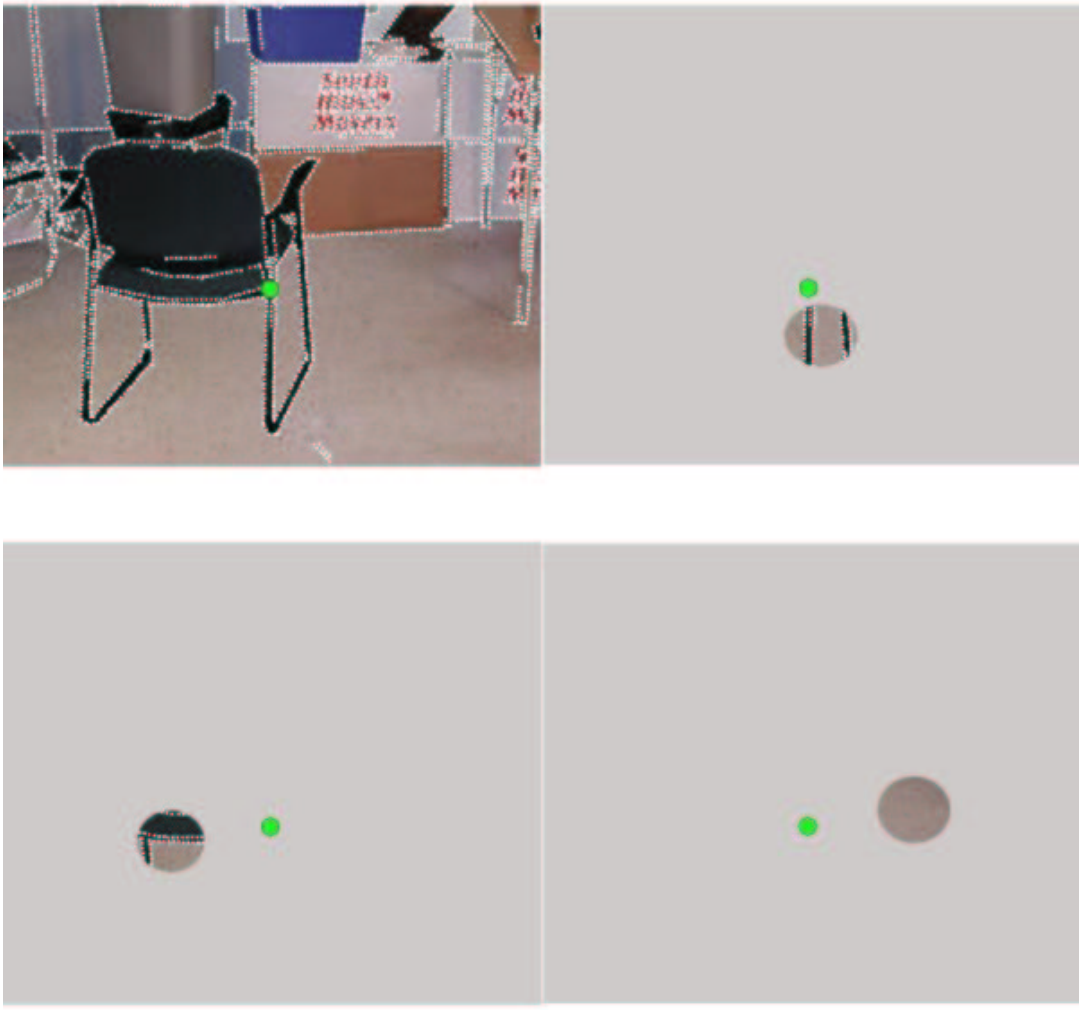


Figure 3.5: An illustration of using image probes to classify an edge pixel. The point to be classified is shown in green; to classify the point we evaluate the density of edges in sparse image locations (the circles).

object edge pixels inside the rectangle surpasses a threshold for some image region, we decide that an instance of the target object is in that region. Figure 3.1, fourth column, shows example aggregation results, and Figure 3.9 illustrates an aggregation filter applied to an image.

In other words, for each pixel on each edge, we examine a local edge grouping surrounding it. How edges are arranged inside the aperture, termed the *local edge configuration* for the pixel, is the cue used to determine whether the pixel belongs to the object or the background. We train a classifier from example views to discriminate local edge configurations of clutter edge pixels from those of object edge pixels. Unfortunately, if the aperture is too small, the local edge configuration may be ambiguous; in other words, it might be impossible to tell which class the edge point belongs to based on edge information inside the aperture. For this reason, ambiguous edge points are passed on to a second classification phase, which considers the local edge configuration in a larger aperture. If it is still unclear at this stage whether the edge point belongs to the foreground or background, we attempt to classify it based on features in a larger aperture, and so on. As an illustration, Figure 3.4 depicts four phases in this cascade process for the recognition of the chair in the lower left portion of the image. Additionally, Figure 3.7 shows a block diagram of how images are processed. We classify edge pixels based on a series of growing apertures, rather than one large aperture, for efficiency reasons; that is, classifying edge pixels based on information in larger apertures is computation-intensive, so we prefer to only spend that computation on edge pixels that are ambiguous at that aperture size.

At each phase in the cascade, a decision tree computes a sparse set of localized image features which measure edge density in some image neighborhood. The locations of the image features are determined according to a tree structure which is learned at training time. Figure 3.6(c) illustrates the classification of one edge point, at one classification phase.

We focus on part classification because a rich literature of prior research exists on aggregation of individual edge pixels into overall object instances, for example [46][54]. These techniques break down in high clutter, so an approach to part classification which effectively isolates the object edge pixels from the background edge pixels would boost the usefulness of these prior aggregation techniques.

Section 3.5 describes our algorithm in general. In Section 3.6 we present a set of experiments which demonstrate the viability of the technique for the recognition of three everyday wiry objects. Additionally, Section 3.6 addresses concerns related to the sensitivity and complexity of our algorithm.

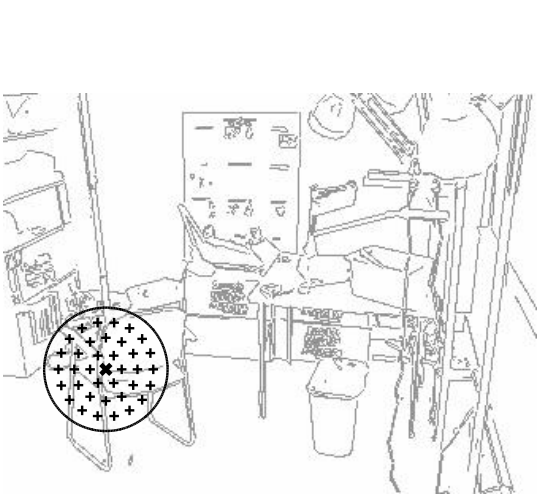
3.4 Assumptions and Capabilities

This section briefly summarizes the major assumptions that underlie our technique and the capabilities it provides. First, we assume that it is possible to reliably detect binary edges in the image which correspond to the geometric structure of our target objects. Specifically, this assumption implies that lighting and object material properties behave in such a way that lighting-related edges (caused by specularities, highlights, and so on) are rare. We further assume that we have access to many labeled training images (like Figure 3.3) showing the target objects in a variety of poses. In terms of object pose, we assume that object rotation in the plane of the image is small, and that variation in object scale over all images is small. We note, however, that since our target objects are constrained by gravity they will not generally appear at an arbitrary in-image-plane rotation. Also, we note that it is possible to extend our technique to handle object scale variations either by processing the same image repeatedly at a variety of scales [105] or by rectifying image features to a canonical scale [71][34]. Since our approach heavily processes large amounts of training data, we assume access to large amounts of memory and CPU time.

Our algorithm enables the localization of wiry objects at arbitrary out-of-image-plane rotation in large amounts of clutter. Moreover, our algorithm is capable of localizing the target object despite deviations between training and run-time images in terms of arrangement of the clutter objects, camera viewpoint, rotation of the target object, and (to a degree) the room or environment the images were taken in. Finally, our technique requires the precise estimation of few arbitrary parameters compared to previous approaches.

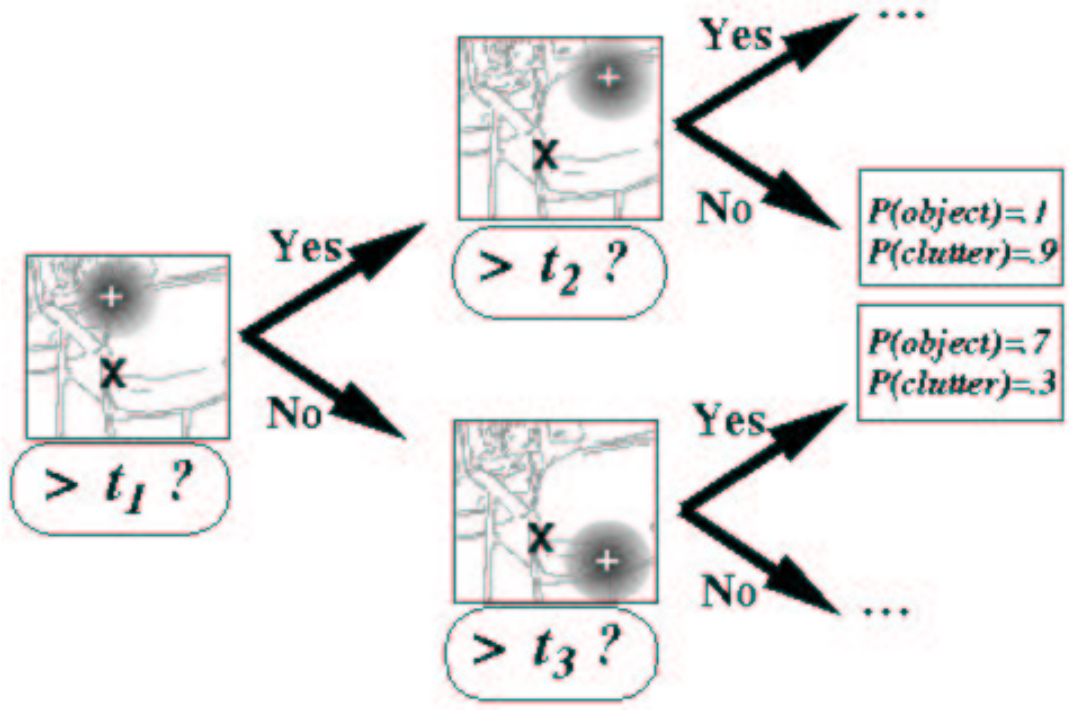
3.5 Approach

Our approach to edge filtering consists of training a cascade of classifiers which discriminates object edge pixels from background edge pixels based on local edge density features. Section 3.5.1 defines the features used to train the classifiers, and Section 3.5.2 describes how classifiers trained from these features are applied to the image in a cascade format. Section 3.5.3 discusses the classifiers themselves.



(a) Aperture and shifted probe centers

(b) An edge probe



(c) Classification of a query edge point

Figure 3.6: Overview of one phase of the recognition cascade. 3.6(a) Edge probes are evaluated in an aperture surrounding a query edge point. The query edge point is marked "X," and edge probes are evaluated at locations marked "+."3.6(b) Each edge probe measures edge density in some image neighborhood. Here an edge probe is evaluated at shifted probe center $\mathbf{q} + \delta$. for a query edge point q. 3.6(c) Edge points are classified by evaluating edge probes according to a tree structure.

3.5.1 Image Features: Edge Probes

As stated above, we assume that our objects are well-characterized by their shape properties, and that the shape properties are represented by binary edges extracted from the image. For this reason, we classify each query edge pixel based on the values of edge features which summarize the local shape in image neighborhoods in its immediate vicinity.

Specifically, an *edge probe* at *probe center* $\mathbf{p}$ over a list of edge pixels $\mathcal{G}$ is defined as

$$ep(\mathbf{p}, \mathcal{G}) = \sum_{\mathbf{t} \in \mathcal{G}} \exp\left(-\frac{\|\mathbf{p} - \mathbf{t}\|^2}{\sigma^2}\right)$$

where $\mathbf{t}$ and $\mathbf{p}$ are 2-vectors of $[x, y]$ image coordinates. An edge probe can be thought of as a Gaussian receptive field with variance σ^2, centered at point $\mathbf{p}$ in an edge image whose edge pixels are contained in $\mathcal{G}$. Edge probes measure the density of edge pixels in some neighborhood in the image; in this sense, each edge probe is analogous to a bin in a shape context histogram [9]. The variance σ^2 is a user-set parameter; however an experiment reported in Section 3.6.4 suggests that our edge filtering results are not highly sensitive to its setting. See Figure 3.6(b) for an illustration of an edge probe evaluated at probe center $\mathbf{q} + \delta$.

Edge probes summarize the local shape in image neighborhoods in terms of the density of edges in those neighborhoods. By computing edge probes at a set of image neighborhoods surrounding a query edge pixel $\mathbf{q}$, we summarize the overall shape properties of the image near $\mathbf{q}$, which should be a discriminating recognition cue for our shape-based objects.

Specifically, for each query edge pixel $\mathbf{q}$, we compute edge probes at probe centers in the spatial neighborhood of $\mathbf{q}$ to classify it as object or clutter. Consider a set of *relative probe centers* $\Delta = \{\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \cdots, \boldsymbol{\delta}_k\}$, $\boldsymbol{\delta}_i = [x_{\boldsymbol{\delta}_i}, y_{\boldsymbol{\delta}_i}]$, laid out over a 2D grid centered at the origin. To classify $\mathbf{q}$, we shift the relative probe centers so that they surround $\mathbf{q}$, and compute edge probes $ep(\mathbf{q}, \Delta, \mathcal{G}) = \{ep(\mathbf{q} + \boldsymbol{\delta}_1, \mathcal{G}), \cdots, ep(\mathbf{q} + \boldsymbol{\delta}_k, \mathcal{G})\}$ at *shifted probe centers* $\{\mathbf{q} + \boldsymbol{\delta}_1, \cdots, \mathbf{q} + \boldsymbol{\delta}_k\}$. An illustration is shown in Figure 3.6(a)-3.6(b).

Given a fixed σ, we space the relative probe centers at intervals of σ pixels so that they evenly blanket a circular aperture as in Figure 3.6(a). We arrange the probe centers this way so that every pixel in the aperture is able to contribute to edge probes evaluated at one or more relative probe centers in a local neighborhood. But how large should the aperture be? We want the shifted probe centers to cover a large enough neighborhood that the edge

probes will contain sufficient information to discriminate object pixels from background pixels. At the same time, however, if the aperture is too large, an intractable amount of computation will be required at training time to evaluate edge probes that might not be crucial for classification. Worse, if the aperture is so large that most of the edge probes are totally irrelevant to the category of the query edge pixel, error-prone classifiers could be trained[59][1]. Thus, we are presented with "the aperture problem" which appears in many computer vision problems– when attempting to induce information about a particular location in the image we want to incorporate image data from a large enough surrounding area, but not so large that we introduce irrelevant data or useless computation.[1]

3.5.2 Part Classification: The Cascade

We solve our aperture problem in phases– we first identify those edge points whose class is discriminable based on very nearby features, then identify points that are made discriminable by features slightly farther away, and continue to do so until the aperture covers the entire object in question. The overall objective for the cascade is to maximize the number of true positive pixels and minimize the number of false positive pixels. As an illustration, Figure 3.4 shows the results of classification at four phases of the cascade, and Figure 3.7 gives a block diagram showing the structure of the cascade.

In addition to providing a solution to our aperture problem, the classifier cascade allows fast screening of edge pixels whose classification is easily determined based on information in a small window, leaving the bulk of the computation to more ambiguous sections of the image. Similar cascades have recently achieved significant speedups for template-based approaches to recognition– see Section 6.2 for a discussion of them.

More formally, consider a set of relative probe centers Δ which cover a circular aperture as in Figure 3.6(a). Define $r(\Delta)$ to be the radius of the circle. Our approach is to train a series of classifiers $\{c_1, c_2, \cdots, c_k\}$ which evaluate edge probes at sets of relative probe centers $\{\Delta_1, \Delta_2, \cdots, \Delta_k\}$ such that $\{r(\Delta_1) < r(\Delta_2), \cdots, < r(\Delta_k)\}$ The first classifier in the series, c_1, is trained to classify edge pixels based on edge probes evaluated in a small radius surrounding them; c_2 classifies based on edge probes computed over a slightly larger radius, and so on. Edge pixels labeled "object" by c_1 are classified by c_2; points labeled

[1]We emphasize that there is a critical distinction between the aperture of an edge point (denoted by the black circle in Figure 3.6(a)) and the spatial support of a single edge probe (denoted by the gradient-shaded region surrounding $\mathbf{q} + \boldsymbol{\delta}$ in Figure 3.6(b))– the aperture describes the image region over which *all* edge features for a given query point are evaluated, while the portion of the image which contributes to a *single* edge feature is determined by the edge probe support.

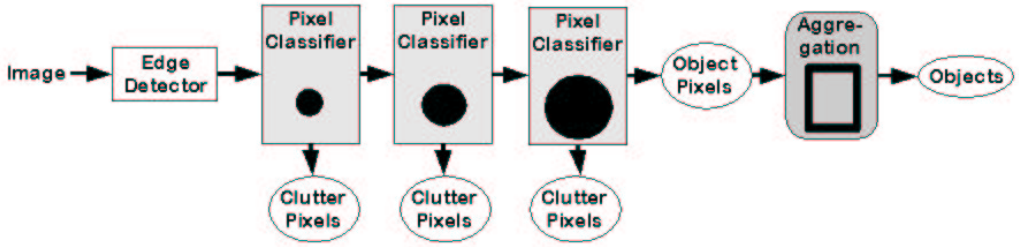


Figure 3.7: A block diagram showing the organization of our one-sided classifier cascades as described in Section 3.5.2.

"background" by c_1 are discarded. Edge points labeled "object" by c_2 are passed to c_3, and so on. A block diagram of this cascade design is illustrated in Figure 3.7.

Note that our cascade removes edge pixels from consideration in a *one-sided* way; that is, edge pixels labeled "clutter" with high confidence are removed from the cascade sequence, while all other edge pixels– including those labeled "object" with high confidence– continue to be classified by subsequent phases. An alternative approach is a *two-sided* cascade in which each phase places edge pixels into one of three categories: "object," "clutter," and "uncertain." The "uncertain" pixels are classified by future cascade phases; the "clutter" edge pixels are discarded; and the "object" pixels are immediately added to the set $\mathcal{O}$ of object edge pixels without being further classified by later cascade phases. Two-sided cascades have the computational advantage of removing more edge pixels from classification earlier on, since they remove both object and clutter edge pixels from consideration at each phase. Most of the classifier cascades recently presented in the literature are one-sided (for example, [118][68][125]); an example of a two-sided cascade is [69].

We build one-sided cascades for two reasons. One is that they are simpler to train; essentially, training each phase in a one-sided cascade involves estimating a single threshold between classifier scores for "object" and "clutter" pixels, while each phase in a two-sided cascade requires the estimation of two thresholds which separate the the classifier scores into the three categories (*i.e.*, "object," "clutter," and "uncertain"). The second reason is that we assume that the number of clutter edge pixels to be classified is extremely high relative to the number of object edge pixels. This means that the computational burden of repeatedly classifying object edge pixels by all cascade phases is insignificant relative to the computation required to filter out the clutter edge pixels. Since object edge pixels are relatively "rare events," [125] we disregard the slight inefficiency of repeatedly classifying them, and focus instead on filtering out the high number of clutter edge pixels.

3.5.3 Part Classification: Decision Trees

We seek classifiers $\{c_1, c_2, \cdots, c_k\}$ which compute a sparse set of edge probes over apertures $\{\Delta_1, \Delta_2, \cdots, \Delta_k\}$. The classifiers assign each of the edge pixels in a set of training images to object or background categories; the objective of the classifiers is to minimize the number of false positive pixels, and maximize the number of true positive pixels, among the training images. Our assumption is that classifiers that do so will generalize well. In other words, we assume they will achieve high rates of true positive pixels and low rates of false positive pixels when applied to novel views of the target object and backgrounds.

We choose decision tree classifiers [91] for their ability to quickly classify based on small, sparse sets of features. A classifier c consists of a set of nodes connected in a tree structure; each node represents the evaluation of an edge probe at some shifted probe center. To classify a query edge point $\mathbf{q} \in \mathcal{G}$, we start at the root of the tree and evaluate $ep(\mathbf{q} + \boldsymbol{\delta}_1, \mathcal{G})$, where $\boldsymbol{\delta}_1$ is the relative probe center associated with the root of the tree. Depending on whether the value of the edge probe is greater or less than some threshold t_1, we shift to one or the other of the children of the root node, and evaluate edge probe $ep(\mathbf{q} + \boldsymbol{\delta}_2, \mathcal{G})$ where relative probe center $\boldsymbol{\delta}_2$ is associated with the child. We move from node to node based on the evaluation of edge probes until we encounter a leaf node. Associated with each leaf node is a score w which represents the probability that the query edge pixel belongs to the object. The part classifier returns the triple $(\mathbf{q}, l_+, w)$. By setting a threshold on w, we arrive at a binary decision about whether the image point is an object or background pixel. In short, the application of a classifier to an edge pixel consists of a series of edge probes whose probe centers are dependent on the structure of the tree. Figure 3.6(c) illustrates the application of one classifier to one query edge pixel.

The training procedure for decision trees is a two-step process of tree generation and pruning, following the reduced-error pruning approach of Quinlan [91]. In this framework, the training data is split into two subsets, which we will refer to as the *tree-growing set* and the *holdout set*. We use standard tree induction techniques to build a decision tree with high classification accuracy on the tree-growing set; the score associated with each leaf is estimated by counting the number of object edge pixels and background edge pixels classified to the leaf. Then, subtrees are pruned from the tree when doing so improves a performance criterion on the holdout set [91][14][16].

Section 3.5.4 discusses pruning. Tree induction, in more detail, works as follows. We begin by considering all possible binary splits $(t, \boldsymbol{\delta}_1)$ of the training set into two categories $\mathcal{A}$ and $\mathcal{B}$ such that $ep(\mathbf{q}_a, \mathcal{G}_a) < t < ep(\mathbf{q}_b, \mathcal{G}_b) \forall \mathbf{q}_a \in \mathcal{A}, \mathbf{q}_b \in \mathcal{B}$. Several criteria exist for

assigning scores to splits based on how well they partition the training data into categories containing high percentages of object and background pixels respectively (see, for example, [62][18][79]). The split with the highest score is selected for the root of the tree; then, we partition $\mathcal{A}$ and $\mathcal{B}$ by high-scoring splits, and repeat the partitioning of the training examples until no splits with high scores are available. Each leaf of the tree corresponds to a set $\mathcal{S}$ of training examples, composed of a subset $\mathcal{S}_+$ of object edge pixels and a subset $\mathcal{S}_-$ of background edge pixels. The score assigned to the leaf represents the probability that an edge pixel classified to that leaf corresponds to the target object. The score we assign to the leaf is its *Laplace-corrected* empirical probability $(|\mathcal{S}_+| + (c-1))/(|\mathcal{S}| + c)$. We chose this score following [14], which showed that for some problems using Laplace-corrected scores led to higher classification performance than the empirical probability $(|\mathcal{S}_+|)/(|\mathcal{S}|)$. Here, we choose $c = 2$ as in [14], however we note that it is possible to tune c for optimal performance as in [127].

Rather than assign real-valued scores, it is more traditional for the tree induction procedure to have each leaf assign a binary classification to the edge pixels. For example, early approaches to decision tree induction would label all edge pixels classified to a leaf as "object" if $(|\mathcal{S}_+| + (c-1))/(|\mathcal{S}| + c) > .5$. However, our decision tree leaves assign real-valued scores to allow for flexibility at pruning time. Specifically, our pruning procedure described in Section 3.5.4 simultaneously removes subtrees from the decision tree and selects a threshold t such that all leaves with scores $(|\mathcal{S}_+| + (c-1))/(|\mathcal{S}| + c) > t$ classify edge pixels as "object." By modulating t and changing the topology of the tree, the pruning procedure has greater flexibility in optimizing performance criteria than if it were only allowed to change the tree topology. In other words, by postponing binary decisions at leaves until pruning, we allow the pruning procedure to optimize the tree to greater performance.

We chose decision trees based primarily on two desirable qualities. The first is sparsity; decision trees are capable of classifying examples based on a sparse set of features that are selected by the training algorithm at each node. In practice (see Figures 3.14 and 3.15 for example), we find that in most cases a small subset of the total number of possible edge probes are evaluated in order to classify each pixel. The second desirable property is that the framework for training decision trees consists of a discrete optimization framework of adding and deleting sets of nodes. That is, we can phrase the problem of tuning the decision tree to maximize a performance criterion in terms of adding or deleting nodes if doing so increases the criterion. In Section 3.5.4, we grow decision trees so that they discriminate between object and background edge pixels well, and then tune them by deleting nodes when doing so increases a performance criterion related to the relative importance of false

positive pixels and false negative pixels. In Section 4.2, however, we optimize the tree by deleting nodes when doing so increases performance criteria related to overall instances of target objects.

3.5.4 Part Classification: Pruning

As described above, our goal is to train the decision tree classifier to produce high rates of true positive pixels and low rates of false positive pixels. However, as we describe below, optimizing the decision tree often involves a tradeoff between false positives and false negatives. In this section, we describe how to represent this tradeoff mathematically as a *receiver operating characteristic* (ROC) curve, and how to tune decision trees based on the characteristics of ROC curves.

ROC Curves

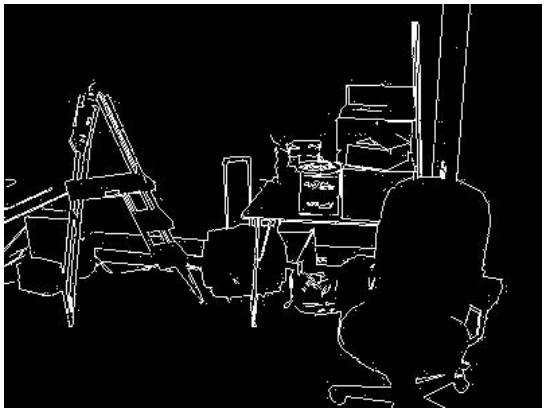
At run time, the decision tree processes each pixel $\mathbf{q}$ in the test image, yielding a triple $(\mathbf{b}, l, w)$ with a real-valued score w. We arrive at a binary classification for the pixel by thresholding w – if w is higher than a threshold t, $\mathbf{q}$ is classified as "object," and if not it is classified "background." Some number of object edge pixels in the test image will have classification scores higher than t; these are the true positive pixels for that classifier and threshold setting. Object edge pixels with scores below the threshold are the false negative pixels since they are falsely classified as background. Background edge pixels with scores higher than the threshold are falsely assigned to the object so they are the false positive pixels; background edge pixels with scores below the threshold are the true negative pixels. Ideally, for a given classifier, we would be able to find a threshold t such that all of the background edge pixels are rejected as clutter and all of the object edge pixels are assigned to the object. That is, we want $t < w$ for all object pixels $\mathbf{q}_+$ with scores $(\mathbf{q}_+, l_+, w)$, and $t > w$ for all background pixels $\mathbf{q}_-$ with scores $(\mathbf{q}_-, l_-, w)$. Unfortunately, due to the various sources of classifier training error– overfitting, ambiguity, weak features, noise– the scores for object and background edge pixels usually overlap to some degree. Because of this, different settings of the threshold t lead to different numbers of false negatives and false positives. Usually setting t involves a tradeoff between false positives and false negatives; for example, if t is set very high, the classifier will be very "selective" so that very few background pixels will have scores $w > t$, meaning that there will be few false positives. However, usually if t is very high there will be many object pixels with scores

$w < t$, meaning the false negative rate will be high.

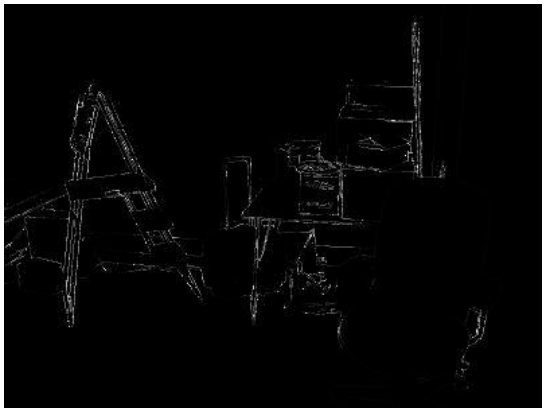
The ROC curve is a way to visualize the tradeoff between false positives and false negatives for a particular classifier. An example is shown in Figure 3.8. Given a particular test image (Figure 3.8(a)), we use a decision tree to assign a score $(\mathbf{b}, l, w)$ to each edge pixel $\mathbf{q}$ (Figure 3.8(b)). Figure 3.8(c) shows a histogram of the scores $\{w_+\}$ for object pixels (in blue) and a histogram of scores $\{w_-\}$ for background pixels (in red). Setting a threshold for the classifier amounts to selecting a vertical line in the histogram at score t; points represented in the bars to the left are classified as background, and points represented in the bars to the right are classified as object. For each such threshold t, some number fp of background pixels and some number tp of object pixels will lie to the right of the threshold, or in other words have higher scores than t. The ROC curve plots the *operating point* (fp, tp) for each possible setting of t. The green and red arrows in Figure 3.8(c) represent two possible settings of t, and their corresponding operating points (fp, tp) are shown in Figure 3.8(d). As such, the ROC curve represents all the possible ways that we can coax the tradeoff between false positives and false negatives by setting the threshold. Figures 3.8(e) and 3.8(f) show which edge pixels have scores higher than the thresholds at the green and red arrows, and illustrate the tradeoff between false positives and true positives in the context of classifying pixels on the ladder. In the former case, most of the pixels on the ladder are retained (*i.e.* the false negative rate is low) but most of the pixels among the background are retained too (high false positive rate). In the latter case, most of the background edge pixels are discarded (low false negative rate) but most of the ladder pixels are thrown away as well (high false negative rate). ROC curves have historically been used to represent this tradeoff in signal processing systems (see, for example, [26]), and more recently the machine learning community has used them to analyze how classifiers perform over a range of threshold settings [90][15].

ROC-based Pruning

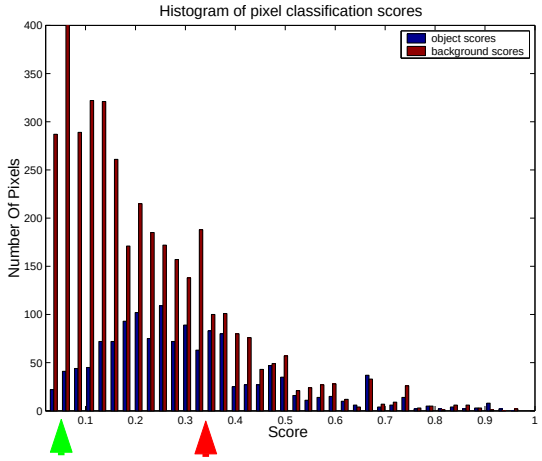
Since ROC curves represent the tradeoff between false positives and false negatives over a range of decision threshold settings, our pruning procedure aims to remove subtrees from the decision tree whenever doing so improves a performance criterion called an *ROC grading criterion* assessed on the ROC curve. The ROC grading criterion reflects the relative importance of false positives and false negatives, which may vary from application to application. For this reason, our pruning framework is general enough to incorporate arbitrary ROC grading criteria.



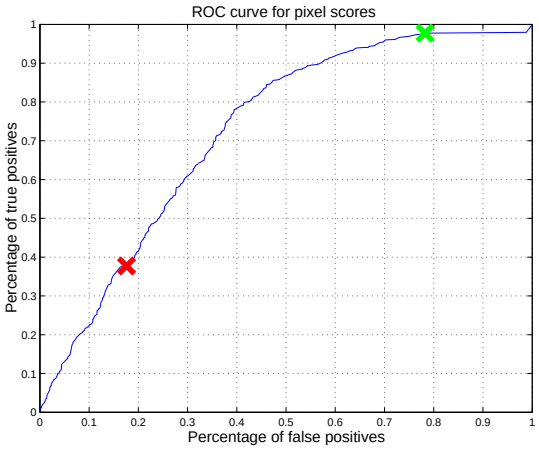
(a) Input edges



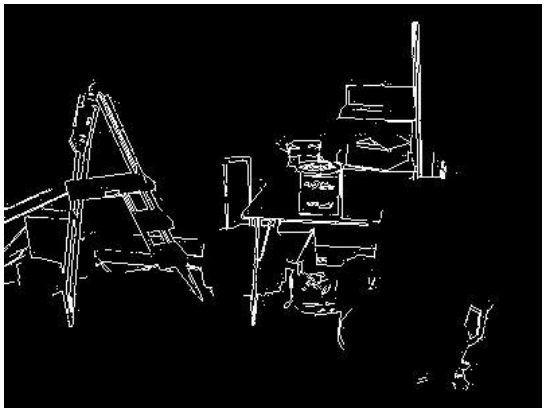
(b) Raw pixel scores



(c) Histogram of scores



(d) ROC curve



(e) Pixels passing low threshold

(f) Pixels passing higher threshold

Figure 3.8: An illustration of using ROC curves to represent classifier characteristics. Figure 3.8(a): An input edge image to be classified by a pixel classifier. Figure 3.8(b): The pixel classifier assigns a continuous score to each edge pixel. Figure 3.8(c): A histogram of all the scores assigned to edge pixels in the image. Red bars correspond to the number of background edge pixels receiving that score, and blue bars represent object edge pixels. The green and red arrows represent two possible thresholds for the classifier: pixels with scores higher than the threshold are classified as "object," and pixels with lower scores are discarded. Figure 3.8(d): An ROC curve represents the tradeoff between false positives and false negatives as the threshold is varied. An operating point (fp, tp) on the curve represents, for a particular threshold, the percentage of background pixels fp above the threshold vs. percentage of object pixels tp above the threshold. Each "X" represents the operating point for the corresponding threshold in Figure 3.8(c). Pixels whose scores are above the low (green) threshold are shown in white in Figure 3.8(e), and pixels with scores above the higher (red) threshold are shown in white in Figure 3.8(f).

Specifically, during pruning, we look at each sub-tree in turn and consider two different classifiers: c_1 consists of the full decision tree, and c_2 consists of a modified tree in which the current sub-tree has been replaced by a leaf node. We classify the holdout images with each classifier and graph the tradeoff between false positives and false negatives for the two classifiers as ROC curves r_1 and r_2. We represent the ROC curves as functions mapping fp to tp. The ROC curves are then evaluated using an ROC grading criterion, $grade$, which assigns a numerical score $grade(r_1)$ and $grade(r_2)$ to r_1 and r_2. If $grade(r_2) > grade(r_1)$, we remove the subtree from the tree; otherwise the tree stays unaltered. After pruning, we invoke a *threshold function* $t = h(r)$ which returns a decision threshold (or, equivalently, an operating point) for the classifier. This process is summarized in Algorithm 1. Our inductive bias is that the ROC curve for the holdout pixels will generalize well to novel test images, so that a particular threshold t will lead to similar operating points on the holdout and test images.

Algorithm 1 Algorithm for pruning a decision tree based on ROC curves.

Require: Set of labeled image features $\{(\mathbf{v}_{l_+}, l_+)\}$ and $\{(\mathbf{v}_{l_-}, l_-)\}$, classifier c, ROC grading criterion $grade$, ROC threshold function h.

for all Subtrees d of c **do**
  $c_d = c \setminus d$
  Compute $\{c(\mathbf{v}_{l_+})\}$ and $\{c(\mathbf{v}_{l_-})\}$
  Compute $\{c_d(\mathbf{v}_{l_+})\}$ and $\{c_d(\mathbf{v}_{l_-})\}$
  $r = roc\{c(\mathbf{v}_{l_+})\}, \{c(\mathbf{v}_{l_-})\}$
  $r_d = roc\{c_d(\mathbf{v}_{l_+})\}, \{c_d(\mathbf{v}_{l_-})\}$
  if $grade(r_d) \geq grade(r)$ **then**
    $c = c \setminus d$
  end if
end for

Compute $h(c)$

As stated above, the ROC grading criterion and the threshold function reflect the performance priorities of the application at hand in terms of false positives and false negatives. Here, since the classifiers we train are modules in an overall cascade, our ROC grading criterion encourages the classifiers to contribute to low false negative and false positive rates for the cascade as a whole.

In particular, consider Figures 3.8(e) and 3.8(f), corresponding to the green and red operating points in Figure 3.8(d), respectively. In the former case, the background pixels

classified as "object" will be passed to further classifiers in the cascade, which may in turn re-classify these pixels correctly. In the latter case, object pixels classified as "background" will be discarded by the classifier and will not be reconsidered by later classifiers. In other words, false positives are recoverable errors, and false negatives are irrecoverable, so false negatives incur a much higher penalty than false positives. For this reason, our threshold function selects operating points to encourage the lowest fp possible while constraining tp to a fixed, high percentage θ of the number of object edge pixels. In so doing, we push the classifier to make progress on the overall cascade goal of having few false positives while constraining it to contribute as few errors as possible – false negatives – that later cascade phases are unable to recover from. This strategy can be thought of as the converse of a Neyman-Pearson criterion [25], which attempts to maximize tp given a fixed threshold on fp. Furthermore, our ROC grading criterion is based on the same fixed detection rate criterion; that is, if (fp, tp) is an operating point for our ROC curve, then the grading criterion is inversely proportional to fp.

Setting the parameter θ requires assumptions about the expected number of phases in the cascade and the expected difficulty in separating object edge pixels from background. On one hand, setting $\theta < 1$ allows the classifier to effectively "give up" on $(1-\theta)\%$ of the most difficult object edge pixels in the holdout set, which gives the classifier a certain amount of robustness to outliers. On the other hand, if the each classifier "gives up" on $(1-\theta)\%$ of the object edge pixels in its training set, then after k cascade phases the true positive rate of the overall cascade will be θ^k. Good values for θ require balancing the need to discard difficult object edge pixels as false negatives, while trying not to degrade the overall cascade true positive rate too much. In Section 4.3 we present two alternative ROC grading criteria and one alternative threshold function.

The training and run-time behavior of our edge-filtering algorithm are summarized in Algorithm 2 and Algorithm 3. Here, for a given set of edge lists $\mathcal{G} = \{\mathcal{G}_1, \cdots \mathcal{G}_k\}$, $c(\mathcal{G}) = \{c(\mathcal{G}_1), \cdots, c(\mathcal{G}_k)\}$, and $c(\mathcal{G}_i) = \{\mathbf{q} \in \mathcal{G}_i \mid c \text{ classifies } \mathbf{q} \text{ as 'object'}\}$. $\mathcal{G}_{i+}$ is the subset of $\mathcal{G}_i$ containing object edge pixels extracted from the image and $\mathcal{G}_{i-}$ contains clutter edge pixels.

For each edge pixel $\mathbf{q}$ in each image in the tree-growing set, we compute edge probes at all shifted probe centers $\{\mathbf{q}+\boldsymbol{\delta}_1, \cdots, \mathbf{q}+\boldsymbol{\delta}_k\}$ corresponding to the relative probe centers $\{\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \cdots, \boldsymbol{\delta}_k\}$ in the smallest aperture Δ_1. Decision tree induction then iteratively splits the edge points into subsets according to the values of edge probes corresponding to selected relative probe centers. Each edge pixel in the holdout set images is then classified by

Algorithm 2 Pseudocode for the training procedure for recognition based on a cascade of edge probes.

Require: Edge lists $\mathcal{G} = \{\mathcal{G}\}$, sets of probe centers $\{\Delta\}$, θ, σ.

Split $\mathcal{G}$ into a tree-growing set $\mathcal{T}$ and a holdout set $\mathcal{H}$.
Let $FP = \{\mathbf{q}_- \in \mathcal{T} \cup \mathcal{H}\}, TP = \{\mathbf{q}_+ \in \mathcal{T} \cup \mathcal{H}\}$.
for all Δ **do** // *loop over cascade phases*

  for all $\mathcal{T}_i \in \mathcal{T}$ **do** // *loop over the tree-growing set*
    for all $\mathbf{q}_+ \in \mathcal{T}_i$ **do** // *loop over object edge pixels*
      $\mathbf{z}_{\mathbf{q}_+} = ep(\mathbf{q}_+, \Delta, \mathcal{T}_i)$
    end for
    for all $\mathbf{q}_1 \in \mathcal{T}_i$ **do** // *loop over clutter edge pixels*
      $\mathbf{z}_{\mathbf{q}_-} = ep(\mathbf{q}_-, \Delta, \mathcal{T}_i)$
    end for
  end for

  Train a decision tree c to discriminate $\{\mathbf{z}_{\mathbf{q}_+}\}$ from $\{\mathbf{z}_{\mathbf{q}_-}\}$

  for all $\mathcal{H}_i \in \mathcal{H}$ **do** // *loop over the holdout set*
    for all $\mathbf{q}_+ \in \mathcal{H}_i$ **do** // *loop over object edge pixels*
      $\mathbf{z}_{\mathbf{q}_+} = ep(\mathbf{q}_+, \Delta, \mathcal{H}_i)$
    end for
    for all $\mathbf{q}_- \in \mathcal{H}_i$ **do** // *loop over clutter edge pixels*
      $\mathbf{z}_{\mathbf{q}_-} = ep(\mathbf{q}_-, \Delta, \mathcal{H}_i)$
    end for
  end for

  Prune c to minimize $|c(\mathcal{H}_-)|$ such that $|c(\mathcal{H}_+)| = (1 - \theta) * |\mathcal{H}_+|$. See Algorithm 1.
  if $|c(\mathcal{H}_-)| < |\mathcal{H}_-|$ **then**
    Add c to cascade $\mathcal{C}$.
    $\mathcal{H} = c(\mathcal{H}), \mathcal{T} = c(\mathcal{T})$ // *discard correctly classified background edge pixels*
  end if

end for

the resulting tree, and subtrees are removed if the pruned tree reduces the number of background edge pixels classified as object edge pixels while keeping the percentage of object edge pixels correctly classified to 1-θ. Edge pixels from the tree-growing and holdout sets classified as object edge pixels by the pruned tree then pass to the training of the second phase in the classifier cascade: for each of these edge pixels, edge probes are computed at all shifted probe centers corresponding to relative probe centers in Δ_2, and so on.

Given a test image, we apply the trained classifiers to each of its edge pixels in turn. An edge pixel classified as "object" by the first classifier is passed to the second classifier; the second classifier classifies the point again, and so on until the point is labeled as "clutter" or the cascade ends.

Algorithm 3 Pseudocode for the run-time evaluation of a novel image based on a cascade of edge probes.

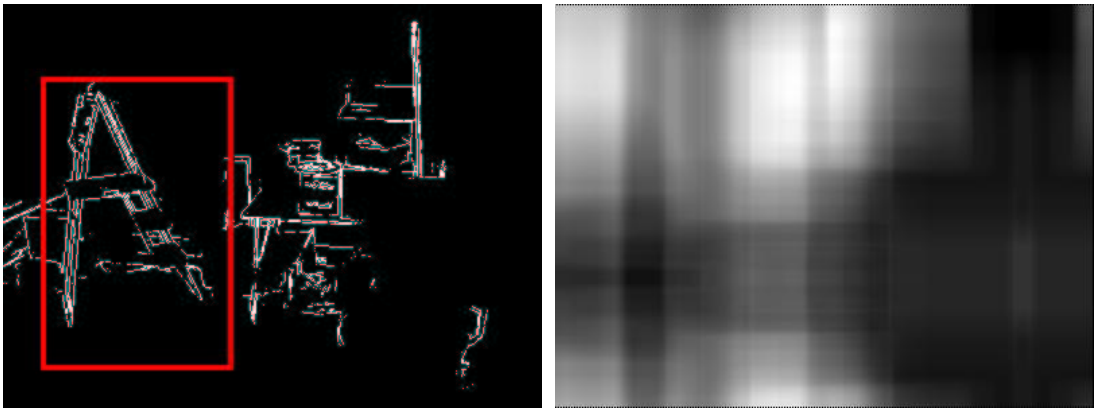
Require: List of edge pixels $\mathcal{G}$
1: $\mathcal{O} = \mathcal{G}$
2: **for all** $c \in \mathcal{C}$ **do** // *loop over classifiers in cascade*
3: $\mathcal{O} = c(\mathcal{O})$ // *discard background edge pixels*
4: **end for**
5: Return $\mathcal{O}$

3.5.5 Aggregation

To give an example of how filtered edge images may be used as an input to aggregation processes, we implemented a simple filter called an *aggregation filter* which scans the filtered edge image with a rectangular template roughly the size of the target object. At each image location, the number of edge pixels falling inside the template is recorded, and image locations with a large number of edge pixels inside the template are noted as likely locations of the target object.

The justification for this aggregation scheme is its simplicity at training and run time. While aggregation filters do not reason about how the pixel scores are distributed spatially (as in standard template-based approaches to recognition, for example [105]), they are extremely fast to apply at run-time since they sum over regions of the filtered edge image. Furthermore, "training" an aggregation filter consists of estimating a single parameter: an acceptance threshold t on the number of pixels inside the rectangle.

We first generate a rectangular template by computing the bounding boxes of object edge pixels in each training image, and averaging the size of those bounding boxes. This



(a) Input edges (b) Aggregation scores

Figure 3.9: An illustration of aggregation by adding the number of points in a box. The pixels marked in white in Figure 3.9(a) were classified positively by the first phase of the classifier cascade. The red box depicts the size of the aggregation filter scanned over the image. Figure 3.9(b) represents the aggregation scores for each image location. The intensity of each pixel represents the aggregation filter score for a box whose upper lefthand corner is anchored at that pixel.

gives us a characteristic bounding box $b_{\boldsymbol{\theta}} = [b_{\boldsymbol{\theta}w} b_{\boldsymbol{\theta}h}]$ whose width b_{1w} and height b_{1h} represent the mean size of the object in the training images. Next, we run each training image through the cascade of edge probes described above. Each training image results in a set $\{(\mathbf{v}, l_+, w)\}$ of features $\mathbf{v}$ identified with the target object. Alternatively, for each training image, consider a binary image B such that $B[\mathbf{q}] = 1$ if $\mathbf{q}$ is classified as belonging to the object, and $B[\mathbf{q}] = 0$ otherwise. We apply an *aggregation filter* to each location in these binary images:

$$ag(b_{\boldsymbol{\theta}}, \mathbf{q}, B) = \sum_{x \in [\frac{-b_{\boldsymbol{\theta}w}}{2}, \frac{b_{\boldsymbol{\theta}w}}{2}]} \sum_{y \in [\frac{-b_{\boldsymbol{\theta}h}}{2}, \frac{b_{\boldsymbol{\theta}h}}{2}]} B[\mathbf{q}[x] + x, \mathbf{q}[y] + y]$$

if $ag(b_{\boldsymbol{\theta}}, \mathbf{q}, B)$ is high, the aggregation step reports the presence of an object instance centered at $\mathbf{q}$ and covering a box of width $b_{\boldsymbol{\theta}w}$ and height $b_{\boldsymbol{\theta}h}$ in the corresponding training image. The set of aggregation scores for target object and background portions of the training images give rise to another ROC curve, this one used to determine a threshold on $ag(b_{\boldsymbol{\theta}}, \mathbf{q}, B)$ so that a high number of true positive *boxes* are reported– that is, image locations $\mathbf{q}$ such that a bounding box centered there with size $[b_{\boldsymbol{\theta}w}, b_{\boldsymbol{\theta}h}]$ covers most or all of the target object– with a minimum of false positive boxes. Figures 3.1 and 3.13 show test images which used an aggregation filter whose threshold was set by hand. In Chapter 4 we present experiments in which the aggregation filter threshold was set using ROC analysis.

We emphasize that this particular aggregation procedure is in no way an optimal proce-

Figure 3.10: Left: Example of scanning an aggregation filter for the ladder over a raw edge image which has not been filtered to remove clutter edge pixels. Right: result of scanning the same edge density filter over the edge image after filtering. See Section 3.5.5 for details.

dure for recovering an object-level description of the contents of the image; in particular, a variety of alignment algorithms[5][117] would be able to give a more precise correspondence between the test image and a reference image or 3D model. Still, this experiment illustrates that the edge filtering procedure can be a useful preprocessing step to higher-level edge-based recognition processes that may fail in extreme clutter. For example, Figure 3.10 shows a test image for which the aggregation filter fails when applied to the raw edge image, but is able to localize the object once the clutter edges have been removed by the classifier cascade.

Table 3.1 summarizes all of the operating parameters that control the behavior of our overall algorithm. Some parameters are set manually by an end user, and others are estimated by an automated procedure during training. Note that in Section 3.6.4, we show that while σ is set manually by the user, recognition performance is relatively stable to a range of settings. Also, an extension to the algorithm, presented in Section 4.1, relaxes the requirement that edge detector parameters and σ be set by hand.

3.6 Experiments

To validate our approach we address the problem of detecting three common objects in highly cluttered indoor scenes under high variation of out-of-image-plane rotation. We evaluate the performance of our edge filtering procedure by computing the true positive rate and false positive rate in each image. We emphasize that, since we represent objects at a pixel level, “true positive rate” does not mean “percentage of times the object was

Parameter	Notation	How Set	Where Defined
Image Features			
Edge Detector Parameters		**Manually:** Usually by trial and error.	Section 3.5.1
Edge Probe Support	σ	**Manually:** By visual inspection of shape features in images.	Section 3.5.1
Relative Probe Center Placement In An Aperture	Δ	**Automatically,** given σ. The probe centers are σ pixels apart along a circle.	Sections 3.5.2, 3.6.2
Aperture Sizes	$r(\Delta)$	**Automatically,** given σ. Rings of probe centers σ pixels apart in radius are added to the aperture.	Sections 3.5.2, 3.6.2
Part Classification			
Laplace Correction Number	c	**Manually** set to 2.	Section 3.5.3
Fixed Detection Rate Threshold	θ	**Manually** set to enforce very low fn_p	Section 3.5.4
Aggregation			
Aggregation Filter Size	$[b_{\boldsymbol{\theta}w} b_{\boldsymbol{\theta}h}]$	**Automatically** from object bounding boxes in training images	Section 3.5.5
Aggregation Count Threshold		**Automatically** from ROC analysis of aggregation counts on training images	Section 3.5.5

Table 3.1: This table lists the various parameters which control the behavior of the cascade of edge probes algorithm described in Chapter 3.

detected." Instead, it means "percentage of object *pixels* detected." Thus, even if the true positive rate is below 100%, it may still be possible to conclusively locate the object in all test images, since the density of object edge points will be relatively high at the true location of the object if the true positive rate is relatively high. Likewise, "false positive rate" does not mean "number of times a section of the background was mistakenly labeled as the object," but rather to "number of times an *edge pixel* in the background was mistakenly labeled as belonging to the object." Thus, even if the false positive rate is greater than zero, it may be possible to achieve zero false detections of the object, especially if the falsely-detected background edge pixels are sparsely distributed in the scene. Indeed, as explained in Section 3.5.5 and indicated in Figures 3.2 and 3.13, the filtered edge image is an encouraging starting point for aggregation of individual pixels into overall localizations of the object in the image.

3.6.1 CROC Plots

We visualize the performance of a cascade of classifiers using *cascade ROC*, or CROC, plots. We classify all pixels **q** in the test images with the first classifier in the cascade and evaluate the performance of the first classifier via its true positive rate tp_1 and false positive rate fp_1. The pixels classified as "object" are passed to the second classifier in the cascade, which classifies them with some true positive and false positive rates tp_2 and fp_2, and so on. A CROC plot for the cascade plots $\{(fp_1, tp_1), (fp_2, tp_2), \cdots, (fp_k, tp_k)\}$ for each of the classifiers in the cascade. CROC plots succinctly summarize the contribution made by each of the cascade phases toward the final goal of discarding all false positives while retaining all true positives. A label k (or F_k) next to a point in a CROC plot represents that the point corresponds to the kth classifier in the sequence.

3.6.2 Chair And Cart

For our first set of experiments we took 150 images each of the cart and chair against a blue screen (Figure 3.12(a)). The images span the full revolution of the objects in the plane parallel to the floor. The elevation of the camera varies by approximately 25 degrees with respect to the object, and the extent of scale variation across images is about 10%. Note that for some of our images, the generic viewpoint assumption is violated[40]; for example two of the legs of the chair in Figure 3.12(a), second row, fourth column, are accidentally aligned.

We also took images of a background environment consisting of a set of "office" objects– for example lamps, a table, and boxes(Figure 3.12(b)). The set of views spans roughly 60 degrees of rotation in the plane parallel to the floor, and variation in scale and camera elevation is about the same as for the cart and chair images. To induce appearance variation in the background between views, we modified the poses of each background object and shuffled their relative positions every 5 to 8 images. The camera was moved between each view.

The images used as training and testing data for the chair and cart are composites of random pairs of foreground and background images (Figure 3.2, bottom row, and Figures 3.12(a)-3.12(b) show examples). The pairing was done without replacement; in other words, there is no repetition of foreground or background images in the composite images.

As explained in Section 3.5.2, we wish to address the aperture problem by training a sequence of classifiers $\{c_1, c_2, \cdots, c_k\}$ which evaluate edge probes according to sets of relative probe centers $\{\Delta_1, \Delta_2, \cdots, \Delta_k\}$, each of which evaluates edge probes over increasingly larger image regions. We chose a straightforward way to implement this idea; specifically, we arrange the relative probe centers as a set of concentric rings propagating outward from the origin, and make Δ_n contain the first n rings. Since we want each image location in the aperture to contribute to edge probes at one or more probe centers, we position the relative probe centers in the rings so that together, their spatial supports cover the entire aperture. Specifically, we space the relative probe centers in each ring at intervals of σ pixels along the circle, and space the rings σ pixels from each other in radius. Other layouts for spatial shape features are possible; for example, in [9] they are more densely concentrated near the query edge pixel. We chose a uniform spacing of probe centers for simplicity– specifically, our scheme prevents us from having to make further design choices relating to how the density of probe centers should vary with respect to distance from the query edge pixel. Figure 3.6(a) shows the relative probe center layout for Δ_3. After inspecting the typical shape features in training images of the chair and cart, we set σ to 10 pixels so that the edge probes would capture interesting shape variations in the data without being too redundant. In Section 3.6.4, however, we report recognition results over a range of σ values.

To induce the decision trees, we employ a series of algorithmic techniques which performed well in previously published comparative studies. First we discretize the edge probe values using the implementation of minimum-entropy discretization [31] in the MLC++ software library [61]. We chose to do so following the results in [24] which demonstrated

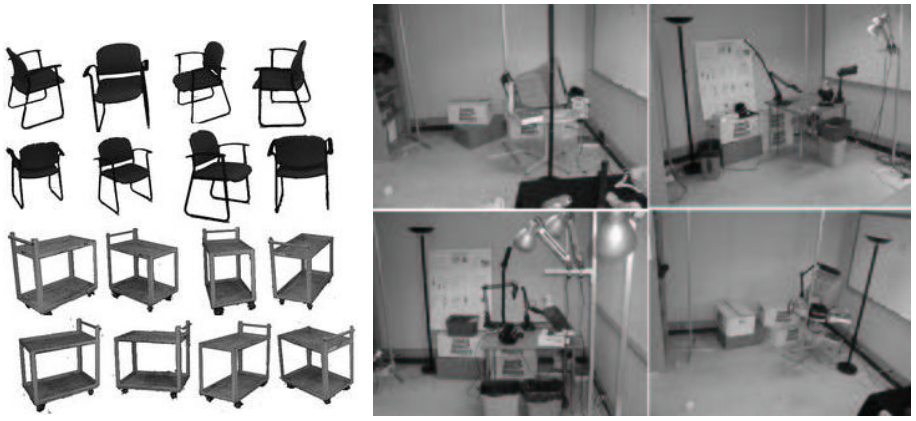
that discretizing continuous training data, especially using entropy-based methods, led to significant classification accuracy increases in trees trained on UCI data sets [11]. After discretization we grow the trees using the ID3 induction routine in MLC++. We used the information gain criterion to generate splits (See Section 4.1.1 for a discussion). While many other splitting criteria have been presented, for example [62][18], we chose information gain because it is the widely-employed standard. Furthermore, we found anecdotally that the one-sided split criteria in [18] generated trees that were completely eliminated during holdout. As mentioned in Section 3.5.3, we assigned Laplace-corrected probabilities at each leaf. We did so following the results in [14], which suggest that pruning with Laplace-corrected leaf probabilities led to increased classification performance on 10 UCI data sets. Anecdotally, we found that the Laplace correction and discretization both had a significant impact on the performance of our decision trees. We used the holdout set to prune the decision trees as described in Section 3.5.3, setting our target false negative rate θ to 2% to constrain each tree in the cascade to classify 98% of its true positive cases correctly.

For each recognition trial on the chair and cart, 150 composite images were partitioned into a tree-growing set of 50 images, a holdout set of 50 images, and a test set of 40-50 images. Edges were detected on all images using the Vista line finder [89]; for computational speed we sample the detected edges at 5 pixel intervals and classify the edge samples. In each image, the ratio of the number of background edge pixels to foreground edge pixels is approximately $10:1$.

A set of sample results on the chair is shown in Figure 3.11, and an example of pixel classification over a series of cascade phases is shown in Figure 3.4. The images show classification results after the first, fifth, tenth, and fifteenth cascade phase. Note that as successive classifiers are applied, using larger apertures, the number of background edge points is reduced dramatically while retaining a high number of edge points on the chair. Note that in both figures, false positives are so sparse and isolated after edge filtering that they are can be easily removed by an aggregation filter mechanism as described in Section 3.5.5 and shown in the fourth column of the figure.

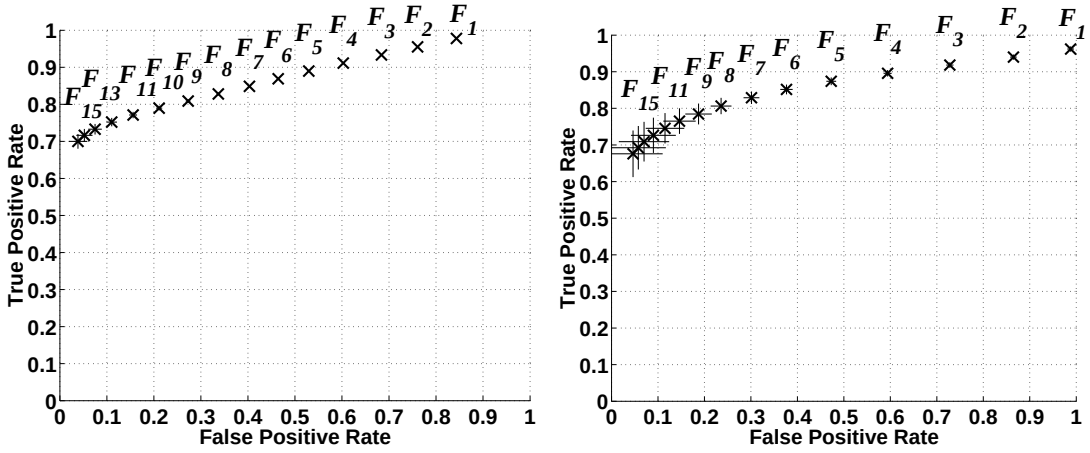
The performance of each tree in the cascade, over all test images containing the chair, is summarized in the CROC plot in Figure 3.12(c). We performed 7 recognition trials; each trial consisted of randomly partitioning the images into tree-growing, holdout, and test sets, training the cascade, and evaluating the number of true positive and false positive pixels as more phases are added to the cascade. More specifically, for each cascade we compute tp_p and fp_p for each test image and recognition trial. The "x" in the graph plots

Figure 3.11: Many example results of edge pixel classification on the chair. The first and third rows show input edges, and the second and fourth rows shows the final classification results.



(a) Foreground images

(b) Background images



(c) ROC curve for the chair

(d) ROC curve for the cart

Figure 3.12: Example foreground and background images and ROC plots for recognition of the cart and chair. See Section 3.6.2.

$(mean(fp_p), mean(tp_p))$; bars extend to the left and right by $var(fp_p)$ and up and down by $var(tp_p)$. Figure 3.12(d) shows an analogous graph for results of 6 recognition trials with the cart. For both objects, the results for $\sigma = 5$ pixels and $\sigma = 20$ pixels are similar. The true positive and false positive rates for the two objects are comparable– for example, roughly 70% of edge pixels on the object are retained, versus 5% false positives among the background.

3.6.3 Ladder

In order to test this technique on real-world images of an object in a variety of clutter settings with no bluescreening, we took 1157 1600-by-1200 images of a ladder in 7 different indoor environments: a classroom, conference room, office, lab, living room, warehouse,

and kitchen (Figures 3.1 and 3.13). For each image, the camera was approximately 3m away from the objects in the scene; the elevation of the camera varied between 1.6m and 1.75m; the set of all images of a particular scene covered about 60 degrees of rotation with respect to the scene objects in the plane parallel to the floor. The camera was moved between each view, and once every five views the ladder was rotated to an arbitrary angle with respect to the ground and the poses and configurations of clutter objects were randomly modified. The depth of the ladder with respect to the camera varied by a total of approximately 20% across all views. Edges were detected in these images using Vista as in the above section, and edges were hand-labeled as belonging to the ladder, or to the clutter objects.

For each experiment, we selected images for tree-growing and holdout sets, and trained cascades of classifiers to filter out background edge pixels as described above. For the experiments in Sections 3.6.3 and 3.6.3, we set the edge probe variance parameter σ to 20 pixels by visual inspection of the shape features present in the images, as in Section 3.6.2. To constrain the decision trees to keep a high true positive rate, we set the pruning parameter θ to 2%. Depending on the arrangement of edges in the training images, it is possible that the decision tree trained for a particular cascade phase may not significantly reduce the number of false positives on the holdout set; therefore, for our experiments on the ladder we skipped a cascade phase if it failed to reduce the false positive rate by 5% or more.

Training And Testing In A Single Environment

First, we considered training individual classifier cascades for each indoor environment separately. For each environment, we randomly split the set of all images of the object in that scene into a tree-growing set of 60 images, a holdout set of 60 images, and a test set containing the remainder of the images. Then the procedure described in Section 3.5 was used to train a 20-phase classifier cascade, and run each of the test images through the resulting edge filter. We used 20 cascade phases because during training, classifiers after the 20th phase failed to reduce the false positive rate significantly. For each test image, we measured the true positive and false positive rates after the final cascade phase. Results are summarized in Table 3.2; examples are shown in Figures 3.1 and 3.13. As in the experiments on the chair and cart, each classifier cascade retained a high percentage of edge points on the object (roughly 71%-78%), while discarding most background edge pixels (roughly 90%).

Figure 3.13: Results of edge filtering and edge grouping for cascades trained to recognize the ladder in various environments. Top to bottom: the kitchen, the lab, the cubicle, the classroom, the conference room, the warehouse, the apartment. Left to right: input images with detected edges overlaid, results of the edge filtering operation described in Section 3.5, and results of edge grouping as described in Section 3.5.5.

Background	# train	# test	mean TP	mean FP	var TP	var FP
Classroom	120	48	0.778	0.102	0.008	0.008
Kitchen	120	43	0.712	0.111	0.038	0.038
Cubicle	120	54	0.718	0.080	0.039	0.039
Conf. Rm	120	63	0.775	0.089	0.018	0.018
Warehouse	120	51	0.755	0.094	0.021	0.021
Living Rm	120	54	0.779	0.064	0.017	0.017
Lab	110	12	0.739	0.210	0.010	0.010

Table 3.2: Edge pixel filtering results for classifier cascades trained to detect the ladder in individual environments. Each row corresponds to edge filtering results for a classifier cascade trained on images taken in the environment indicated in the first column. The second column indicates the total number of images in the tree-growing and holdout sets together. The third column gives the number of independent test images used for evaluation. The fourth and fifth columns give the average true positive (TP) and false positive (FP) rates over all test images, and the sixth and seventh columns give the variance in true positive and false positive rates across all test images. See Section 3.6.3 for further details.

Distinct Training and Testing Environments

To suggest that the performance of our classifier cascades degrade gracefully according to the deviation between training image characteristics and test image characteristics, we trained a cascade on a set of images of the ladder in five of the rooms (kitchen, cubicle, warehouse, living room, and lab), and tested it on images of the other two rooms (classroom and conference room). We randomly selected a total of 120 images from the set of all images of the object in the training environments, using 60 of them for the tree-growing set and 60 for the holdout set. A classifier cascade was constructed and applied to all images of the object in the test environments, i.e. environments *not* present in the training data. Edge pixel classification results are summarized in Table 3.3, using the same notation as Table 3.2. Comparing the corresponding lines in Tables 3.2 and 3.3, we see that filtering performance decreases slightly in some aspects: the true positive rate on the classroom images drops somewhat when the classroom images are not present in the training data, and the false positive rate for the conference room images increases when the conference room images are absent from training. However, some decrease in performance is to be expected when conditions in training and test images vary significantly; the key point is that edge filtering performance degrades gracefully with these variations.

3.6.4 Sensitivity

The edge probe variance parameter σ is a free parameter that critically affects the size of the spatial support region for our basic edge features. In the experiments reported in previous

Background	# test	mean TP	mean FP	var TP	var FP
Classroom	169	0.661	0.072	0.050	0.050
Conf. Rm	183	0.779	0.110	0.013	0.013

Table 3.3: Edge pixel filtering results for classifier cascades trained to recognize the ladder in a different set of environments than those found in the test images. See Table 3.2 for an explanation of notation and Section 3.6.3 for more information.

Sigma	mean TP	mean FP	var TP	var FP
15	0.827	0.094	0.009	0.009
20	0.775	0.089	0.018	0.018
25	0.715	0.074	0.017	0.017
30	0.775	0.125	0.014	0.014

Table 3.4: Edge pixel filtering on the conference room image set for various settings of σ. See Section 3.6.4.

sections, σ was set by hand to make the spatial support of edge probes large enough to integrate edge pixels from small neighborhoods on the target objects. However, it is natural to wonder how the performance of our approach depends on the choice of σ. To address this issue, we trained a set of 4 classifier cascades on the images of the ladder in the conference room (Figure 3.13, fourth row), corresponding to σ values of 15,20,25, and 30 pixels. As in Sections 3.6.3 and 3.6.3, the relative probe centers were arranged in concentric rings at distances $\sigma, 2 * \sigma, \cdots$ pixels from the origin. However, in order to increase the aperture size at a similar rate for all values of σ, we added two rings of relative probe centers per cascade phase for $\sigma = 15$. 60 images of the conference room were randomly selected as a tree-growing set, 60 images made up the holdout set,and the remaining 63 images were used for evaluation. Figure 3.4 summarizes the results, using the same notation as described in Table 3.2. While the true positive and false positive rates do vary across settings of the parameter, in each case the filter retains a high percentage of edge pixels on the object (roughly 71% to 83%) while removing roughly 90% of all background edge pixels. In subsequent experiments, variations in tp_p and fp_p of this magnitude did not significantly affect aggregation performance. Exactly how the setting of σ affects true positive and false positive rates is complex and depends on characteristics of the viewing environment, the objects present, the classifiers in the cascade, and the policy for arranging the relative probe centers in the aperture. Nonetheless, these results suggest that the classifier cascade performs well over a range of reasonable values for this parameter.

An alternative to selecting a setting of σ by hand is to evaluate many different edge probes, corresponding to many different σ settings, at each relative probe center.

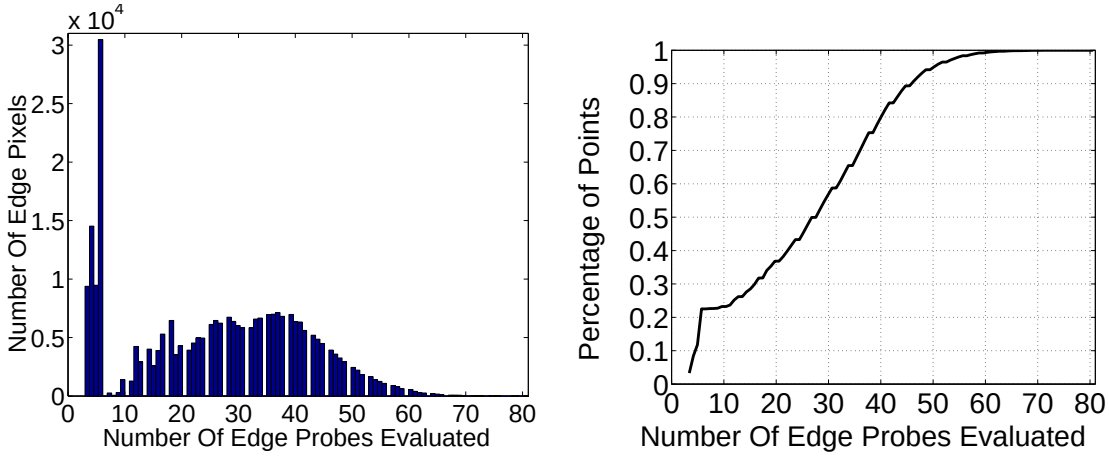


Figure 3.14: Left: Histogram of the number of edge probes evaluated per edge point in 54 test images. Right: Cumulative distribution function for this histogram.

3.6.5 Empirical Complexity

This section addresses concerns about the time and space complexity of our edge filtering approach, in terms of experimental results on the training and test phases. In particular, we demonstrate that the amount of computation required to train the cascade and evaluate it on a novel image is reasonable, despite the potentially large number of image features and classifiers in the cascade.

Training the cascade of classifiers involves computing edge probes and inducing a decision tree for each phase of the cascade. At a particular cascade phase, an exhaustive set of edge probes is computed over all edge points in the training set; thus, space requirements at training time will be determined by the number of edge pixels in the training set at each cascade phase, along with the number of relative probe centers in the aperture at each phase. In Table 3.5 we show these numbers for a few phases of a classifier cascade trained on images of the ladder in the classroom background (Figure 3.13, third row) as described in Section 3.6.3. This table also shows approximate running times of the decision tree inducer on a 1.67 GHz Athlon for those cascade phases, and the number of nodes in the learned decision trees. On one hand, due to our strategy of evenly spacing the relative probe centers in the aperture, the dimensionality of the training data increases as the aperture is grown; on the other hand, since we filter training examples out of the training set at each phase of cascade training, the number of training examples decreases as the aperture is grown. The total time required to train one classifier cascade on a 1.67 GHz machine, including all decision tree induction and edge probe calculation, is approximately one day.

Phase	# features	# examples	time	# nodes
1	6	316338	0:04	896
5	92	213282	0:16	94
10	340	141120	0:44	10
15	746	111674	1:05	11
20	1309	107848	1:20	8

Table 3.5: Number of relative probe centers (second column), number of training examples (third column), decision tree induction times (fourth column, hours:minutes), and number of decision tree nodes for various cascade phases for the classroom image set. See Section 3.6.5.

When evaluating a novel image, some number of edge probes are computed at each edge point in the image until either the edge point is filtered out of the image or the last phase of the cascade is reached. Thus, the time complexity of evaluating a novel image will in large part be determined by the number of edge probes required for each of its edge points. To get a sense of the total number of edge probes computed at typical edge points in test images, we took the classifier cascade trained on images of the ladder in the living room (Figure 3.13, sixth row) as described in Section 3.6.3, and counted the number of edge probes required to classify each edge pixel in each test image of the living room, over all cascade phases. Figure 3.14 summarizes these edge probe counts in a histogram and cumulative distribution. Note that roughly 23% of all edge pixels are classified based on ten or fewer distinct edge probes, and 95% of all points in the test images require evaluation of 50 or fewer edge probes. This is significant since the total number of relative probe centers in the largest aperture (Table 3.5, last row) is 1309. Thus, while the training phase selects edge features from a large set of potential features, relatively few of these features are evaluated for any given image point at run time. As an illustration, Figure 3.15 shows the set of all shifted probe centers for the 20th cascade phase at a typical edge pixel, and the set of 65 shifted probe centers at which edge probes were computed during the classification of the point by all phases of the cascade. Note also that since an edge probe essentially consists of convolution of a portion of the image with a small gaussian kernel, it is fast to compute. An initial, non-optimized implementation of our technique evaluates a 1600x1200 image from the ladder set in about 5 seconds.

3.7 Discussion

In this section we make an initial exploration into what features our decision trees compute in order to classify edge pixels. Since decision trees classify training examples by making a

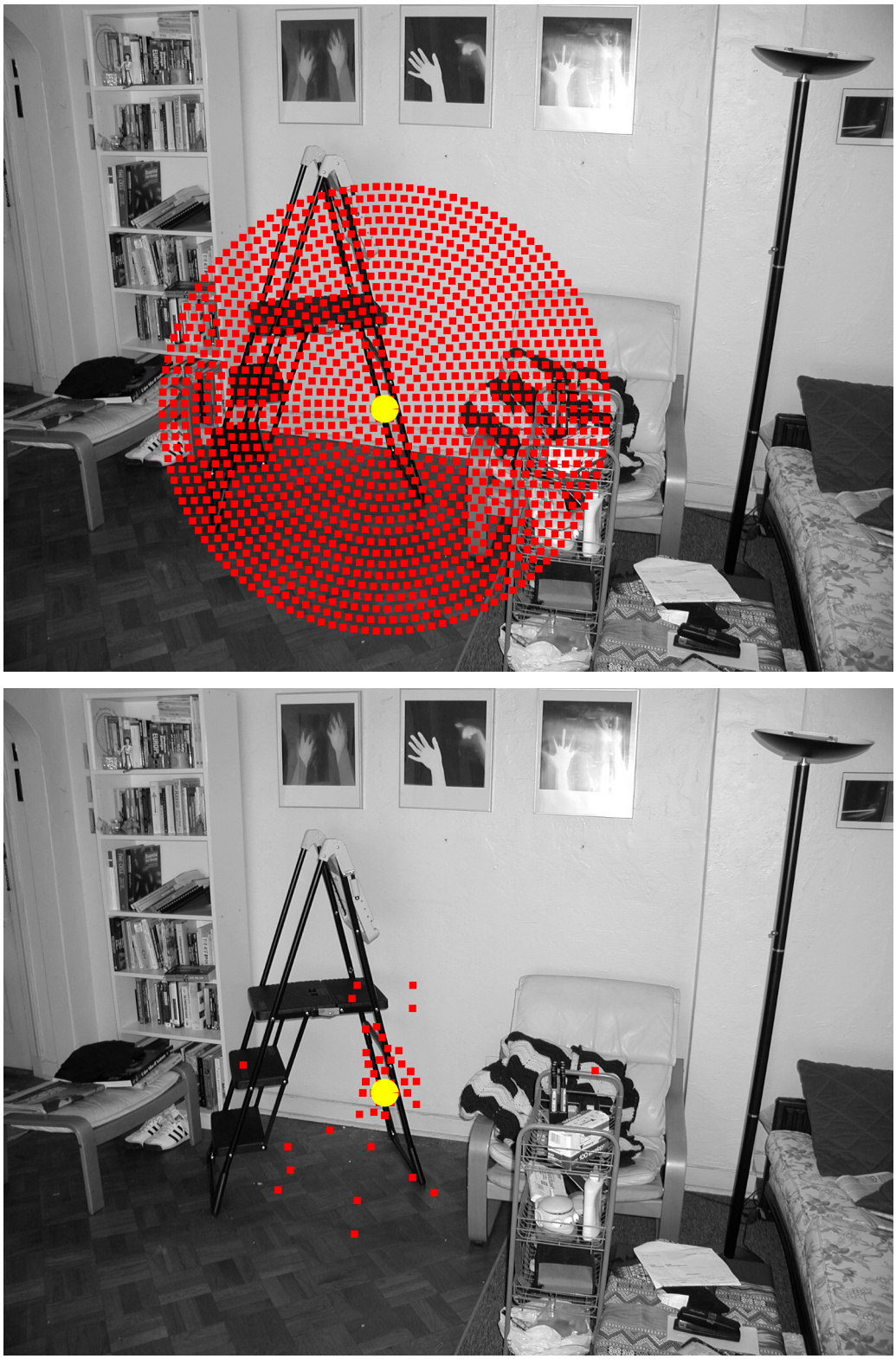
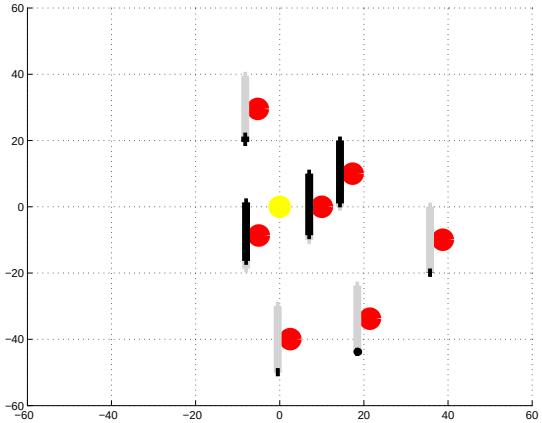


Figure 3.15: Top: The set of all relative probe centers for the 20th cascade phase, shifted to the point at the yellow circle, are shown as red squares. Right: In order to classify the point, edge probes are only evaluated at the probe centers shown in red.

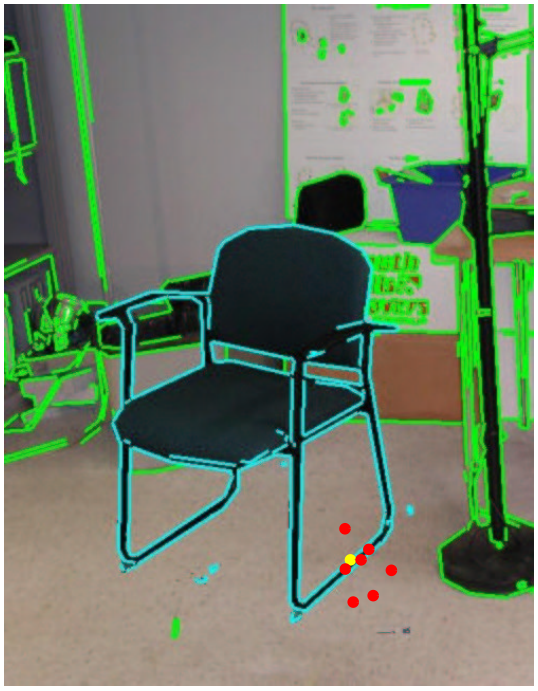
series of threshold tests on individual features, we can traverse the tree in order to interpret it in terms of a step-by-step analysis of what tests it is making in order to arrive at a classification. We stress that our decision trees were not designed at the outset with interpretability in mind; for tree-building techniques along these lines, see for example [18]. However, by manually tracing paths through our trees, we can in some cases gain insight into the visual characteristics being exploited for discrimination. To do so, we examined decision trees induced on the chair data set (Section 3.6.2) and searched for leaves containing large numbers of training examples. For each of those leaves, we traced the path through the tree leading from the root node to the leaf and made a graphical representation of the features being tested at each node and the values of the split.

More specifically, consider representing each node in the decision tree as a pair $(\boldsymbol{\delta}, t)$; a pixel $\mathbf{q}$ is sent down the left branch at that node if $ep(\mathbf{q}+\boldsymbol{\delta}, \mathcal{G}) > t$ and down the right branch of the tree if $ep(\mathbf{q}+\boldsymbol{\delta}, \mathcal{G}) < t$. A path from root to leaf generates a set of these pairs, each pair corresponding to a test at a node along the path. These tests define a set of intervals in which the edge probe values must lie in order for the test pixel to arrive at the leaf. That is, for each relative probe center $\boldsymbol{\delta}$ we can define a set of intervals $\{(t_{min}, t_{max})\}$ such that in order to be classified into the leaf in question, $ep(\mathbf{q}+\boldsymbol{\delta}, \mathcal{G})$ must be greater than t_{min} and less than t_{max}. We represent these valid feature value intervals graphically by plotting the range between the maximum and minimum edge probe values over all edge probes, and the sub-ranges (t_{min}, t_{max}) for each $\boldsymbol{\delta}$. In Figure 3.16, the locations of relative probe centers $\boldsymbol{\delta}$ is plotted in red, the query edge pixel $\mathbf{q}$ is in yellow, the range of all edge probe values is in grey, and the sub-range (t_{min}, t_{max}) is in black.

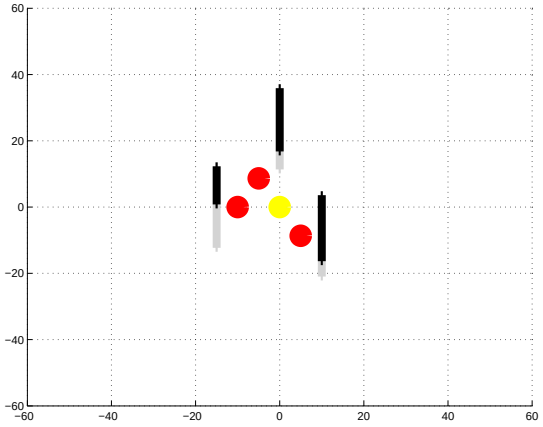
Example displays of admissible feature value intervals for two different decision tree paths are shown in Figure 3.16. In Figure 3.16(a) we show feature intervals corresponding to a highly-populated leaf leading to a positive classification. The display suggests that edge probe values should be in the low to high range of values for probe centers lying on a diagonal line leading from below and left of the point, to above and right of the point. Also, edge probe values should be low for probe centers on either side of that line. This suggests that this particular leaf characterizes edge pixels lying along a diagonal edge that is more or less isolated from the background. Indeed, in several of the training images, the legs of the chair are in this sort of configuration, and we show an example in Figure 3.16(b). Figure 3.16(c) shows admissible ranges of values for edge probes leading to a negative classification. Edge probes evaluated at three probe centers at close proximity above, below, and to one side of the point must be in the mid-to-high range of values in order for the pixel to arrive in the leaf. This suggests that edge pixels with very high edge



(a) Feature values leading to a positive classification



(b) Example image with those features



(c) Feature values leading to a negative classification

(d) Example image with those features

Figure 3.16: Examples of interpreting paths through the decision tree as the selection of ranges of feature values for positive and negative classification. Figure 3.16(a) shows feature ranges corresponding to a path through the tree leading to a positive classification, and Figure 3.16(b) shows an example of an object pixel classified to that leaf. Figure 3.16(c) shows an analogous set of feature ranges for a path through the tree leading to a negative classification, and Figure 3.16(d) shows a background pixel classified to that leaf. The query edge pixel is shown in yellow; relative probe centers used as features in the path to the leaf are shown in red; the range of feasible feature values for those features are shown as the black sub-range over the full range of edge probe values (grey).

density all around them are likely to belong to the background. In fact, most pixels in regions so busy with edges do in fact project to the background (see Figure 3.16(d) for an example).

3.8 Conclusion

Our approach to separating objects from background based on edge cues consists of screening each edge pixel in the image through a series of classifiers, each of which computes sets of edge features over successively larger image areas. Each classifier in the cascade computes a sparse set of localized edge features in a sequence determined by its tree structure. By tuning feature extraction to the object and background present in training images, we overcome the effects of object structure (concavities, holes, wiry structures) which confound template-based approaches to recognition. And by screening the image through a series of increasingly complex classifiers, we quickly discard edge points which are easily discriminated from the object, saving computation for more ambiguous portions of the image.

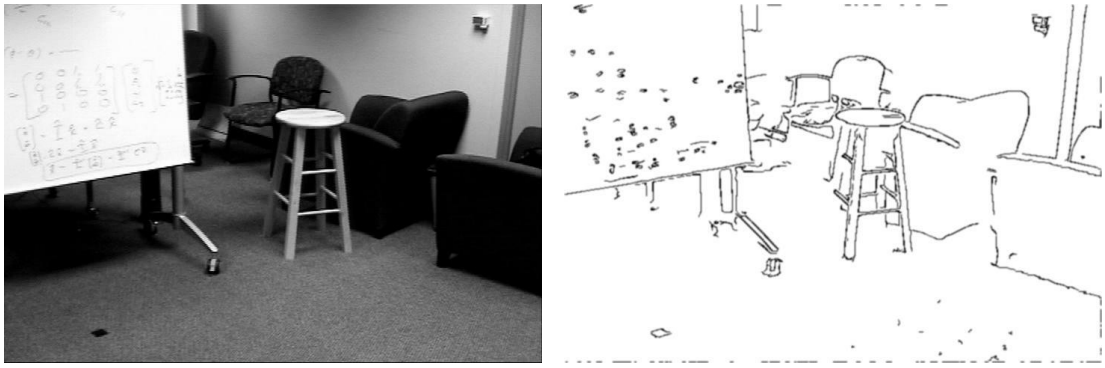
Chapter 4

Extensions

In this chapter we extend the basic algorithm presented in Chapter 3 to enhance its performance in a specific set of recognition scenarios. First, Section 4.1 proposes the use of a richer set of image features to deal with images in which binary edges are difficult to extract reliably. Then, in Section 4.2 we provide a means for adapting our basic part classification algorithm so that part classification and aggregation are tightly coupled. Finally, Section 4.3 gives some more alternatives to fixed detection rate ROC optimization which allows us to adapt our cascade of edge probes to situations in which the relative importance of false positives and false negatives is known before training. Chapter 5 presents the results of experiments which apply these extensions to real image sets.

4.1 Edge Operator Probes

One limitation of the edge-based recognition approach described in the preceding chapter is illustrated in Figure 4.1. Edges are extracted from the image on the left, with the goal of using the configuration of edges to detect the presence and location of the stool at the center of the image. Because some of the edges on the outline of the stool have low contrast (for example, near the bottom of the rear left leg and top of the rear right leg), they are not reported by the edge detector. Since the recognition strategy outlined in Chapter 3 takes binary edge images as input, it will fail to recognize edges present on the object if those edges are not detected by the edge detector. As a result, it is necessary to finely tune the parameters of the edge detector so that all of the edges on the object are detected, and as few spurious edges as possible are detected.



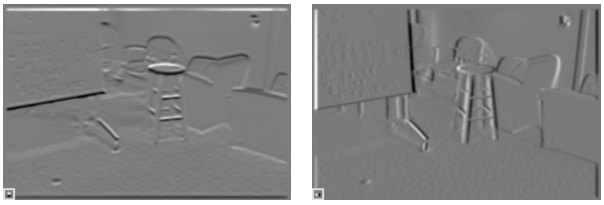
(a) Input Image (b) Extracted Edges

Figure 4.1: An example of binary edge detection failing to capture all the salient edges on a target object. Left : Input image. Right : Binary edges extracted from the image. Note that edges are missing from sections of the rear stool legs.

We address this limitation by making two modifications to the algorithm presented in Chapter 3. First, we re-cast the part classification problem in terms of classifying each pixel in the image, rather than pixels where edges have been detected. Second, for image features we replace edge probes with *edge operator probes*, that is, edge detection operators evaluated at shifted probe centers. In so doing, we tune edge features to a specific set of images and overcome the brittleness of making binary decisions about whether pixels correspond to edges.

For completeness, our new problem statement is as follows. Given a novel image I, our goal is to generate a list of pixels $\mathcal{O}$ such that for each $\mathbf{q} \in \mathcal{O}$, $I[\mathbf{q}]$ projects onto the target object. Our source of training data is a set of example images $\{T\}$ along with lists $\{\mathcal{T}_T\}$, where for each $\mathbf{q} \in \mathcal{T}_T$, $T[\mathbf{q}]$ projects onto the target object. Thus, while in the previous chapter we classified edge pixels $\mathbf{q}$ based on the local arrangement of edges $\mathcal{L}_\mathbf{q}$ in the vicinity of $\mathbf{q}$, here we classify raw image pixels based on edge characteristics of the image patches that surround them.

As in the previous chapter, we wish to extract features from the image which capture the local edge characteristics of an image neighborhood. However, rather than relying on a pre-determined binary edge map for this purpose, we instead wish to apply edge operators directly to local neighborhoods in the raw image. We draw the edge operators from the family of first derivative-of-Gaussian (D-of-G) operators since they are straightforward to parameterize in terms of their scale and orientation characteristics. An *edge operator probe* at probe center $\mathbf{p}$ in image I is defined as



(a) (b)

Figure 4.2: Examples of responses of the image in Figure 4.1(a) to two edge operators. The edge operators are shown at the lower left.

$$eop(\mathbf{p}, I, \boldsymbol{\theta}) = \sum_{\mathbf{t}} g_{\boldsymbol{\theta}}(\mathbf{t}) \cdot I[\mathbf{p} - \mathbf{t}]$$

where $g_{\boldsymbol{\theta}}$ is a first-derivative-of-Gaussian (DOG) operator whose characteristics are controlled by the parameter vector $\boldsymbol{\theta}$. Here, we consider the family of DOG operators parameterized by orientation ϕ and scales σ_x and σ_y in the x and y directions. Specifically,

$$g_{\boldsymbol{\theta}}(\mathbf{p}) = \frac{-x(\mathbf{p}, \phi)}{\sigma_x^2} * \exp(-\frac{1}{2}(\frac{x(\mathbf{p}, \phi)^2}{\sigma_x^2} + \frac{y(\mathbf{p}, \phi)^2}{\sigma_y^2}))$$

and $x(\mathbf{p}, \phi) = \mathbf{p} \cdot \left[\frac{\cos(\phi)}{\sin(\phi)}\right]$ and $y(\mathbf{p}, \phi) = \mathbf{p} \cdot \left[\frac{-\sin(\phi)}{\cos(\phi)}\right]$. The scale and orientation characteristics of $g_{\boldsymbol{\theta}}$ are controlled by the parameters $\boldsymbol{\theta} = [\sigma_x, \sigma_y, \phi]$. An edge operator probe responds strongly to the presence of edges at orientation ϕ and scales described by σ_x and σ_y in the vicinity of pixel $\mathbf{p}$. Examples of two edge operators applied to the image in Figure 4.2(a) are shown in Figure 4.2(b). We choose to focus on this family of edge operators because they are easily parameterized and because they are by far the most common edge operators used as the substrate for edge analysis in practice.

Our strategy will be to select a set of derivative-of-Gaussian operators $\{g_{\boldsymbol{\theta}}\}$, and classify image pixels by evaluating those operators at fixed spatial offsets with respect to them. In other words, the vector of image features used to classify the image patch around $\mathbf{q}$ in image I is $[eop(\mathbf{q}+\boldsymbol{\delta}_1, I, \boldsymbol{\theta}_1), \cdots, eop(\mathbf{q}+\boldsymbol{\delta}_m, I, \boldsymbol{\theta}_1), \cdots, eop(\mathbf{q}+\boldsymbol{\delta}_1, I, \boldsymbol{\theta}_n), \cdots, eop(\mathbf{q}+\boldsymbol{\delta}_m, I, \boldsymbol{\theta}_n)]$, with one entry for each edge operator $g_{\in}\{g_{\boldsymbol{\theta}_1}, \cdots, g_{\boldsymbol{\theta}_n}\}$ evaluated at each relative probe center $\boldsymbol{\delta} \in \{\boldsymbol{\delta}_1, \cdots, \boldsymbol{\delta}_m\}$. By training a cascade of classifiers to correctly classify image pixels based on these image features, we hope to automatically select which edge orientations and scales are relevant for localization of the target object; previously, we had implicitly selected relevant edge characteristics by hand when we tuned the parameters of our binary edge detector.

4.1.1 Feature Selection

There are infinitely many possible edge operators $g_{\boldsymbol{\theta}}$. Therefore, we need to select a small, discrete set of operators to use for our edge operator probes. This set has two requirements: first, they should discriminate object pixels from clutter pixels when evaluated at shifted probe centers; second, for efficiency the set should be as compact as possible, in other words the edge operators should not produce redundant responses on images of interest. This section deals with both issues. Specifically, we address redundancy by initially constructing a pool of edge operators which produce non-redundant responses when applied to training images. Then, prior to training each cascade phase, we select a smaller subset of edge operators to be used for classification in that particular phase; we do so by approximating how useful each operator is for discriminating between the training examples present in that phase.

Before training any classifiers in the cascade, we convolve a representative set of training images with all of the candidate operators and select a subset of operators whose operator responses on the training images are non-redundant. In more detail, let $\Phi = \{\phi_1, \cdots, \phi_a\}$ represent an initial set of orientations sampled uniformly from $[0, \pi]$. Also, let $\Sigma_x = \{\sigma_{x1}, \cdots, \sigma_{xb}\}$ represent an initial set of operator scales sampled uniformly from an interval of reasonable operator scales $[\sigma_{min}, \sigma_{max}]$. Σ_y is an analogous set of scales in the y direction. We start with an initial set of edge operators $\mathcal{G}_1 = \{g_{\boldsymbol{\theta}}\}$ containing one edge operator for each $\boldsymbol{\theta} \in \Phi \times \Sigma_x \times \Sigma_y$. We consider a pair of edge operators $g_{\boldsymbol{\theta}1}$ and $g_{\boldsymbol{\theta}2}$ to be redundant if they produce highly similar operator responses when applied to the training images. We measure the similarity of operator responses by normalized correlation. That is, we convolve each training image T with $g_{\boldsymbol{\theta}1}$ and $g_{\boldsymbol{\theta}2}$, and if $corr(g_{\boldsymbol{\theta}1} \circ T, g_{\boldsymbol{\theta}1} \circ T)$ is higher than a threshold for all T, we consider $g_{\boldsymbol{\theta}1}$ and $g_{\boldsymbol{\theta}2}$ to be redundant edge operators. We reduce the initial set of edge operators $\mathcal{G}_1$ to a smaller set $\mathcal{G}_2$ using the forward selection algorithm shown in Algorithm 4. In short, we individually consider each edge operator $g_{\boldsymbol{\theta}}$ from $\mathcal{G}_1$ in turn, and add it to $\mathcal{G}_2$ only if there are no operators in $\mathcal{G}_2$ with respect to which $g_{\boldsymbol{\theta}}$ is redundant. We emphasize that we are not just correlating the edge operators with each other and discarding edge operators $g_{\boldsymbol{\theta}1}$ when $corr(g_{\boldsymbol{\theta}1}, g_{\boldsymbol{\theta}2})$ is high; rather, we remove edge operators whose responses to the image are highly redundant.

At this point, we are left with a set $\mathcal{G}_2$ of edge operators which span a space of reasonable edge operator parameters but do not produce redundant values when applied to our training images. A reasonable strategy for training a classifier to discriminate between object image pixels and background image pixels is to compute all edge operators in $\mathcal{G}_2$, at all

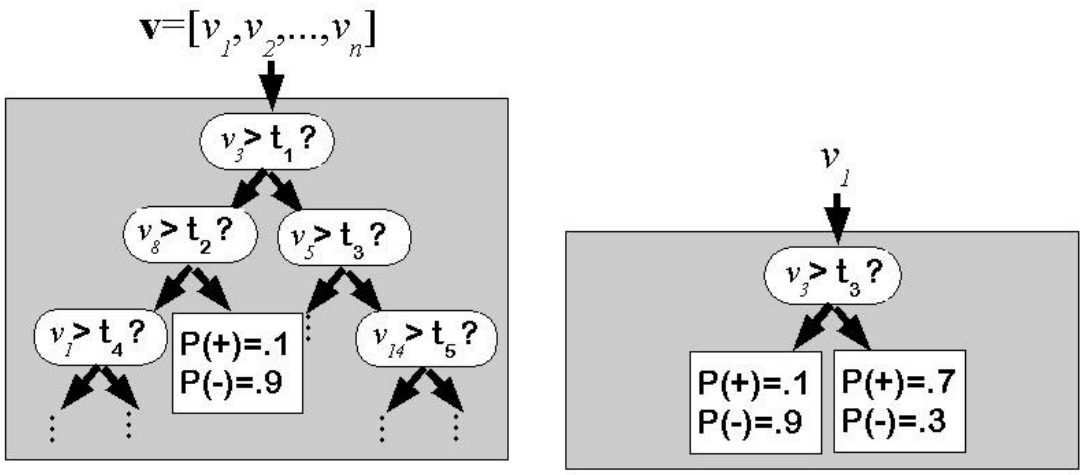
Algorithm 4 **Remove Redundant Operators Selects a subset of edge operators from $\mathcal{G}_1$ which are non-redundant with respect to training images $\mathcal{G}_2$. See Section 4.1.1 for details.**

Require: Set of training images $\mathcal{T}$, set of edge operators $\mathcal{G}_1$, correlation threshold h, candidate operator set $\mathcal{G}_1$.

$\mathcal{G}_2 = \emptyset$.
for all $g_{\boldsymbol{\theta}a} \in \mathcal{G}_1$ **do**
  for all $g_{\boldsymbol{\theta}b} \in \mathcal{G}_2$ **do**
    for all $T \in \mathcal{T}$ **do**
      $c_{ab} = corr(g_{\boldsymbol{\theta}a} \circ T, g_{\boldsymbol{\theta}b} \circ T)$
    end for
  end for
  if $\exists c_{ab} s.t. c_{ab} < h$ **then**
    $\mathcal{G}_2 = \mathcal{G}_2 \cup g_{\boldsymbol{\theta}a}$
  end if
end for
Return $\mathcal{G}_2$

shifted probe centers corresponding to all image pixels and relative probe centers in a set Δ. However, this strategy forces the classification algorithm to learn based on $\|\Delta\| * \|\mathcal{G}_2\|$ features; empirically, the computational load of this large learning problem became unfeasible for reasonable numbers of edge operators and relative probe centers.

To overcome this difficulty, we perform a second stage of feature selection prior to training each classifier in the cascade. Unlike the first feature selection stage, this step is specific to the type of classifier we wish to train. For efficiency, this step is a fast *filter method* [12] which employs a highly simplified analog of the classifier (hereafter the "simple classifier") to make predictions about the usefulness of candidate image features. Generally speaking, filter methods for feature selection train a number of simple classifiers, each of which employ a small number of candidate features to discriminate between positive and negative examples. If a simple classifier discriminates the positive and negative examples well, the corresponding features are assumed to be favorable for training the more complex classifier. The simple classifier is chosen so that training it is very fast; this is what makes filter methods computationally feasible. However, filter methods can fail to assign salient features high scores if the simple classifier is a poor approximation of the more complex one. On the other hand, *wrapper methods* essentially train the full, complex classifier with different feature subsets and pick features whose complex classifiers discriminate the training examples well. Wrapper methods are able to more accurately approximate the utility of each feature because the utility is based directly on the complex classifier as opposed to an approximation of it; however, since wrapper methods imply repeated training of the complex



(a) Complex classifier

(b) Simple classifier

Figure 4.3: Illustration of complex and simple classifiers in the context of feature selection. We select features to use for training the full tree (Figure 4.3(a)) by training a set of stumps (Figure 4.3(b)) and evaluating how well they discriminate the training examples.

classifier, they can be computationally burdensome. We should note that the decision trees we use as part classifiers do their own type of feature selection, in that they select features at each node to split on. For a more in-depth description of feature selection strategies, see [12].

In our case, we train a decision tree at each cascade phase, so we use a simplified decision tree called a *decision stump* as our simple classifier for feature selection (Figure 4.3). A decision stump is simply a decision tree with one decision node; it takes a single feature as input, and classifies instances by thresholding that feature. At each cascade phase, we generate one decision stump per *operator-offset pair*, that is, each possible pair $(g_{\boldsymbol{\theta}}, \boldsymbol{\delta}) \in \mathcal{G}_2 \times \Delta$. For each such operator-offset pair, we compute sets of edge operator probes $\{eop(\mathbf{q}_+ + \boldsymbol{\delta}, T, \boldsymbol{\theta})\}$ and $\{eop(\mathbf{q}_- + \boldsymbol{\delta}, T, \boldsymbol{\theta})\}$ for object pixels $\mathbf{q}_+$ and background pixels $\mathbf{q}_-$. A decision stump is trained to find a threshold which discriminates $\{eop(\mathbf{q}_+ + \boldsymbol{\delta}, T, \boldsymbol{\theta})\}$ from $\{eop(\mathbf{q}_- + \boldsymbol{\delta}, T, \boldsymbol{\theta})\}$; the operator-offset pair is given a score based on how well the decision stump discriminates. The k operator-offset pairs with the best scores are selected as the features used to train the full decision tree.

We score each decision stump using the same criterion used to determine splits during tree induction: namely, for each possible threshold setting t, we compute the so-called "information gain" $gain(t)$ for that threshold setting (see *e.g.* [79]), and set the score for the decision stump to $\max_t gain(t)$. In detail, consider a threshold t which splits a set $\mathcal{S}$ of feature values into two subsets, $\mathcal{S}_l = \{v \in \mathcal{S} | v < t\}$ and $\mathcal{S}_g = \{v \in \mathcal{S} | v > t\}$. The

information gain for this feature and threshold is

$$gain(t) = entropy(\mathcal{S}) - \frac{|\mathcal{S}_l|}{|\mathcal{S}|} * entropy(\mathcal{S}_l) + \frac{|\mathcal{S}_g|}{|\mathcal{S}|} * entropy(\mathcal{S}_g)$$

where

$$entropy(\mathcal{S}) = -\frac{|\mathcal{S}_+|}{|\mathcal{S}|} * \log \frac{|\mathcal{S}_+|}{|\mathcal{S}|} - \frac{|\mathcal{S}_-|}{|\mathcal{S}|} * \log \frac{|\mathcal{S}_-|}{|\mathcal{S}|}$$

and $\mathcal{S}_+$ and $\mathcal{S}_-$ are the subsets of $\mathcal{S}$ that correspond to positive and negative examples (*i.e.* object and background pixels) respectively. Information gain measures the difference between the entropy of $\mathcal{S}$ and the average entropies of $\mathcal{S}_l$ and $\mathcal{S}_g$. While other decision stump criteria do exist (see for example [62][18]), information gain is still widely considered the standard criterion.

Our motivation for using a filter method based on decision stumps is that since we select features using the decision tree splitting criterion, we at least know that the features we select discriminate well as root nodes in the tree. Our assumption is that as decision tree induction recursively partitions the tree-growing set, the features will continue to discriminate well.

Table 4.1 summarizes the image feature and part classification parameters that must be manually set or automatically estimated to use edge operator probes for localization. Since aggregation is not affected by the change in image features and part classification, we omit those parameters in the table.

4.2 Aggregation-Sensitive Part Classification

In the previous chapter we presented part classification and aggregation as two modular steps to be optimized separately. In particular, we first trained a cascade of classifiers to determine which image pixels project onto the object and which project to the background; afterward, we classified the training images with the cascade and used the classified images to set the acceptance threshold of a density filter screening the image for object-sized groupings of positively classified pixels. Our intuition is that the performance of the two phases are tightly coupled; that is, the more successful the part classifier is at removing background pixels and retaining target object pixels, the more successful the aggregation step will be at identifying bounding boxes in the image corresponding to the object, and rejecting bounding boxes covering the background.

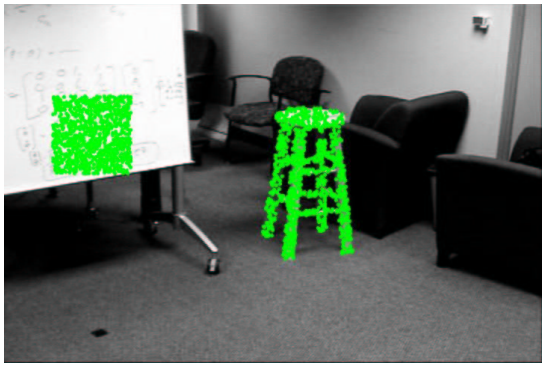
Parameter	Notation	How Set	Where Discussed
Image Features			
Initial Edge Operator Probe Parameters	Φ, Σ_x, Σ_y	**Manually:** They finely cover the space of plausible orientations and scales	Section 4.1.1
Edge Operator Redundancy Threshold	h	**Manually** set to a value close to 1.	Section 4.1.1
Part Classification			
Number of features per classifier	k	**Manually** set to a small value for fast training	Section 4.1.1
Fixed Detection Rate Threshold	θ	**Manually** set to enforce very low fn_p	Section 3.5.4
OR Misclassification Costs	c_{fp},c_{fn}	**Manually** set to strongly encourage very low false negatives and low false positives.	Section 4.3
Relative Probe Center Placement In An Aperture	Δ	**Manually** set so that the operators cover the whole aperture	Sections 3.5.2, 3.6.2
Aperture Sizes	$r(\Delta)$	**Automatically.** Rings of probe centers are added to the aperture.	Sections 3.5.2, 3.6.2

Table 4.1: This table lists the various parameters which control the behavior of cascades of edge operator probes.

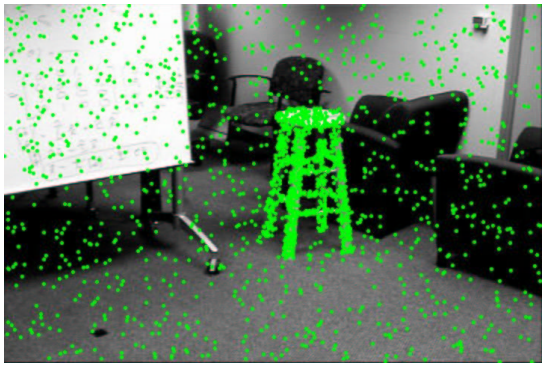
However, it is not necessarily the case that aggregation performance– in terms of the number of true positive bounding boxes tp_b and false positive bounding boxes fp_b found in test images– strictly improves if the number of false positive pixels fp_p decreases and the number of true positive pixels tp_p increases. Specifically, the performance of the aggregation step depends not only on the quantity of true positive and false positive pixels, but also on their spatial distribution. For example, consider a test image whose pixels have been classified by two distinct pixel classifier cascades (Figure 4.4. In both cases, tp_p of the pixels on the target object have been correctly detected, and fp_p of the pixels among the background have been falsely classified as belonging to the target object. However, in the first image (Figure 4.4(a)), all of the false positive pixels lie concentrated in one small portion of the image, while in the second case(Figure 4.4(b)), the false positives are uniformly distributed among the background. If the first image is scanned with an object-sized aggregation filter (Figure 4.4(d)), most image locations among the background will register a score of zero points, and two portions of the image will register high scores– the image section containing the object and the background image section containing the highly-concentrated false positives. Meanwhile, if the second image is scanned with the same aggregation filter (Figure 4.4(c)), each location among the background will register a small, but nonzero, score, while the image region covering the object will register a high score. In the first case, there is potential for confusion between the true-positive portion of the image and the false-positive portion of the image, since the aggregation filter returns high scores for both areas; in the second case, the only section of the image with a high score is the portion containing the true positives, so there is less potential for confusion.

The key point is that the part classifier, by itself, has no capacity to prefer images of the second type over images of the first type, since its only performance metric is based on a function of the sheer number of false positive and true positive pixels, not their spatial distributions. In particular, the cascade training approach presented in the previous chapter greedily adds new classifiers to the cascade if doing so increases a performance criterion based on the number of true positive and false positive *pixels*, but if the end goal of the localization procedure is in fact based on true positive and false positive *boxes*, adding new classifiers to the cascade may actually damage performance.

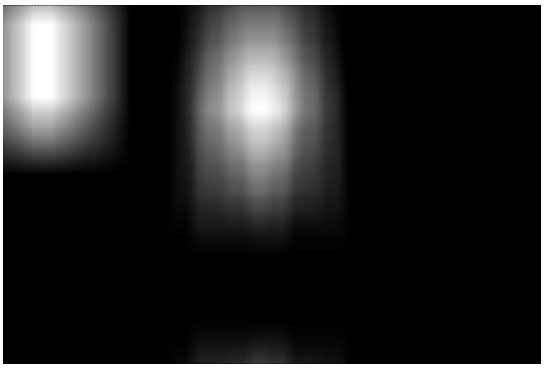
Here, we present a modification to the training procedure presented in Chapter 3 which biases the training of classifiers in the cascade so that criteria based on aggregation performance and pixel-wise classification performance are both optimized. In particular, we modify the decision tree pruning criterion so that sub trees are removed from the decision trees if and only if doing so increases a performance criterion based on true positive and



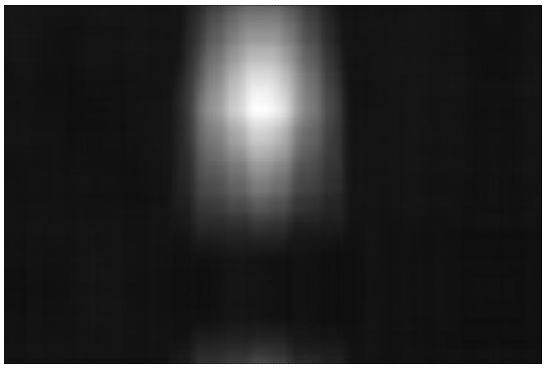
(a) Dense False Positives



(b) Sparse False Positives



(c) Aggregation scores for 4.4(a)



(d) Aggregation scores for 4.4(b)

Figure 4.4: Motivating example for aggregation-sensitive part classification. Figures 4.4(b) and 4.4(a) have the same number of false positive and true positive pixels (in green), but in 4.4(a) the false positives are densely clustered. Figures 4.4(d) and 4.4(c) show the aggregation scores for the two images: it is easy to localize the object based on the points in Figure 4.4(b), while the points in Figure 4.4(a) give rise to a false positive. Unfortunately pixel classifiers have no way to prefer Figure 4.4(b) over Figure 4.4(a).

false positive rates at the box level. We emphasize that the cascade of edge probes presented in the previous chapter can be a useful technique to apply to images on its own, regardless of what sort of aggregation step follows it; in particular, we feel that optimizing the cascade using the technique presented in the previous chapter can be an advantageous pre-processing step prior to registration of the image to a 2D or 3D model. Here, however, we concentrate on the case in which it is known *a priori* that the part classification phase will be followed by a box aggregation procedure. Thus, we refer to this optimization scheme as *aggregation-sensitive part classifier training*.

We begin by slightly modifying the definition of a aggregation filter so that it sums up the raw classifier scores over a rectangular region of the image rather than the number of object pixels in that region. Recall that given a novel image I, we compute image features $\mathbf{v}$ from patches of the image and the part classification step assigns a continuous score w to each image feature. The part classifier thresholds this score and returns all triples $(\mathbf{b}, l, w)$ such that w is higher than the threshold. For each training image, consider the image W, where $W[\mathbf{q}] = w$ is the raw part classifier score given to the image feature $\mathbf{v}$ corresponding to image pixel $\mathbf{q}$. The modified aggregation filter sums these raw scores over an image rectangle covered by the bounding box $b_{\boldsymbol{\theta}}$

$$ag(b_{\boldsymbol{\theta}}, \mathbf{q}, W) = \sum_{x \in [\frac{-b_{\boldsymbol{\theta}w}}{2}, \frac{b_{\boldsymbol{\theta}w}}{2}]} \sum_{y \in [\frac{-b_{\boldsymbol{\theta}h}}{2}, \frac{b_{\boldsymbol{\theta}h}}{2}]} W[\mathbf{q}[x] + x, \mathbf{q}[y] + y]$$

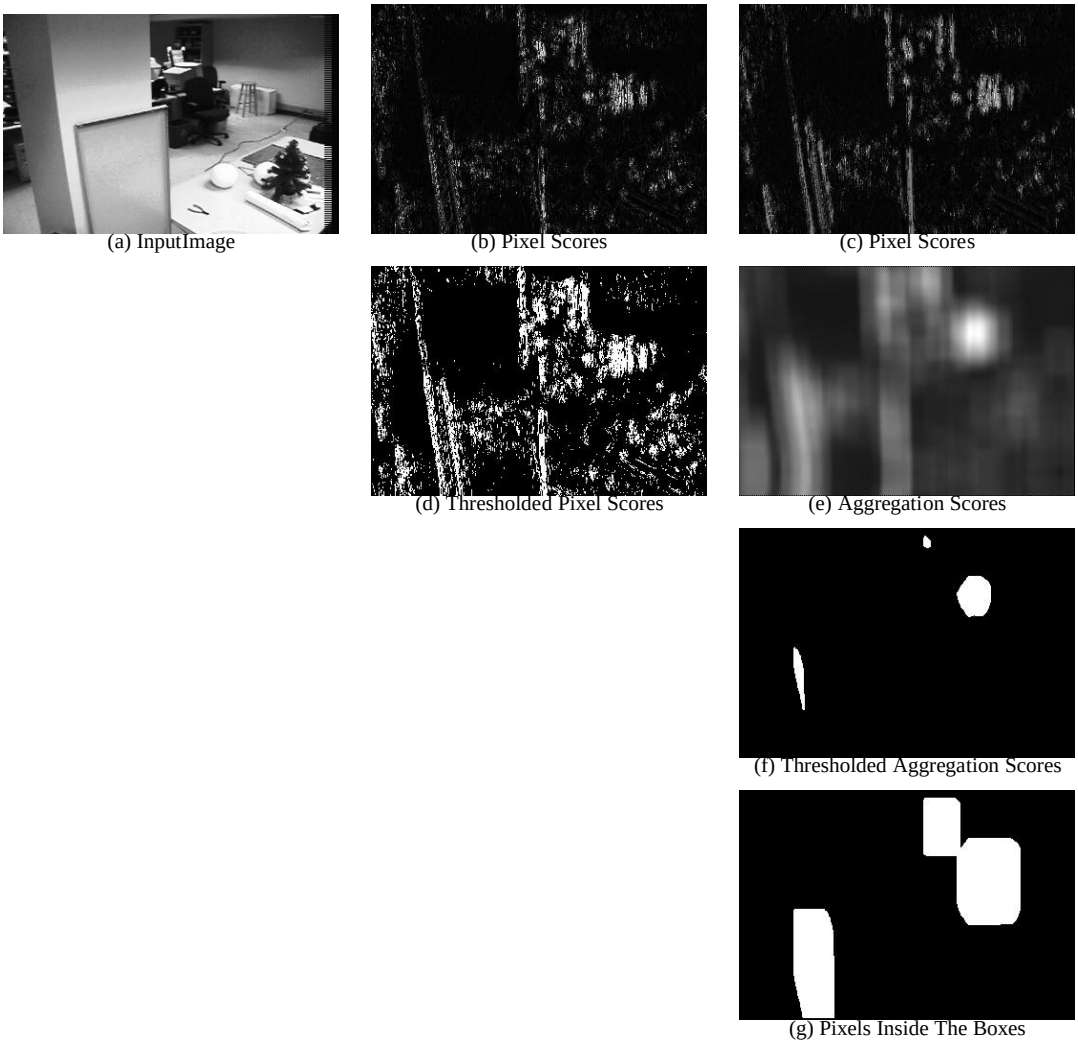
The motivation for this modification is that it allows us to dispense with separate thresholds on pixel classification scores and aggregation scores, so that we can optimize classifiers with respect to a single threshold on aggregation scores. This allows us to make only minor adjustments to our previous pruning scheme, presented in Section 3.5.4, to enable ROC-based decision tree pruning in an aggregation-sensitive way.

As presented in the previous chapter, we grow a decision tree using the training examples in the tree-growing set and then prune the tree using the examples in the holdout set. Here, we follow the same approach of considering each subtree for pruning, and determining whether or not to prune a subtree by using heuristics to grade ROC curves corresponding to the full tree and the pruned tree. Here though, instead of constructing ROC curves based on part classifier scores given to individual image features, we construct ROC curves based on the outputs of aggregation filters evaluated on object and background patches of the holdout images. Specifically, we sample a set of image pixels from each holdout image; these pixels fall into two categories. The first, $\{\mathbf{q}_{l_+}\}$ consists of image pixels near the

center of instances of the target objects in images I. The second, $\{\mathbf{q}_{l\_}\}$, consists of pixels drawn from the backgrounds of those images. For each classifier, we compute the sets $\{ag(b_{\boldsymbol{\theta}}, \mathbf{q}_{l_+}, W)\}$ and $\{ag(b_{\boldsymbol{\theta}}, \mathbf{q}_{l\_}, W)\}$; they constitute the positive and negative examples used to construct the ROC curve for the classifier. In short, previously we constructed ROC curves for classifiers based on the scores that the classifier assigned to object image pixels and background image pixels; here, we construct ROC curves for the classifiers based on the outputs of an aggregation filter applied to object and background portions of the image.

Our modified training algorithm is summarized as follows. At training time we grow a decision tree which discriminates image features corresponding to object pixels in the tree-growing set, from image features corresponding to background pixels in the tree-growing set. For each subtree, and for each image in the holdout set, we produce a pair of images W_1 and W_2, which contain the raw classification scores for those images after classification by the full tree and the tree sans subtree. We compute ROC curves based on the aggregation scores $\{ag(b_{\boldsymbol{\theta}}, \mathbf{q}, W_1)\}$ and $\{ag(b_{\boldsymbol{\theta}}, \mathbf{q}, W_2)\}$, and decide whether or not to prune the subtree based on ROC grading criteria. After pruning, we arrive at a threshold on the aggregation score; any pixels in image regions whose aggregation scores pass the threshold are passed on to the next cascade phase, and all other pixels are discarded. Processing of an individual image is illustrated in Figure 4.5 and contrasted with the standard procedure of classifying images without aggregation-sensitive training. The key difference is that at each cascade phase we aggregate pixel classification scores and use the aggregation scores to determine which pixels pass to the next cascade phase.

Consider how this scheme would handle the earlier example of a classifier which produces false positive pixels finely scattered among the background, versus a classifier whose false positive pixels are concentrated in small regions of the image. In both cases, the false positive pixels would produce relatively high classifier scores. For the first classifier, each of the aggregation filter scores $\{ag(b_{\boldsymbol{\theta}}, \mathbf{q}_{l\_}, W)\}$ corresponding to background portions of the image would be nonzero, but relatively low since any particular rectangle in the background region of the image likely contains few high-scoring background pixels. Therefore, it will be easy to determine a threshold on the aggregation score which rejects most of the aggregation boxes covering background sections of the image, while retaining most of the boxes covering the object. In the second case, most of the aggregation scores for background image locations will be zero, but for the image locations $\mathbf{q}_{l\_}$ covering the concentration of high-scoring background pixels, the aggregation score will be high, making it difficult to determine a threshold which rejects those boxes while retaining those covering the object.



(a) InputImage (b) Pixel Scores (c) Pixel Scores

(d) Thresholded Pixel Scores (e) Aggregation Scores

(f) Thresholded Aggregation Scores

(g) Pixels Inside The Boxes

Figure 4.5: An example of the differences in how images are processed with and without aggregation-sensitive training. The input image is shown in Figure 4.5(a); the second column shows the steps in classifying an image with no aggregation-sensitive training, and the third column shows the steps in classifying an image with aggregation-sensitive training. In the former case, we use a pixel-level classifier to classify each of the pixels in the image (Figure 4.5(b)), and threshold those classifier scores to determine which pixels are discarded (Figure 4.5(d), black pixels) and which are passed on to the next cascade phase (white); in the latter case, we use a pixel-level classifier to classify each of the pixels in the image (Figure 4.5(c)), sum these scores using an aggregation filter(Figure 4.5(e)), threshold the aggregation scores (Figure 4.5(f)), and pass pixels on to the next cascade phase only if they lie inside at least one aggregation box that passed the threshold (Figure 4.5(g), white pixels).

Parameter	Notation	How Set	Where Discussed
Part Classification			
Fixed Detection Rate Threshold	θ	**Manually** set to enforce very low fn_p	Section 3.5.4
OR Misclassification Costs	c_{fp},c_{fn}	**Manually** set to strongly encourage very low false negatives and low false positives.	Section 4.3
Relative Probe Center Placement In An Aperture	Δ	**Manually** set so that the operators cover the whole aperture	Sections 3.5.2, 3.6.2
Aperture Sizes	$r(\Delta)$	**Automatically.** Rings of probe centers are added to the aperture.	Sections 3.5.2, 3.6.2
Aggregation			
Aggregation Filter Size	$[b_{\boldsymbol{\theta}w} b_{\boldsymbol{\theta}h}]$	**Automatically** from object bounding boxes in training images	Section 3.5.5

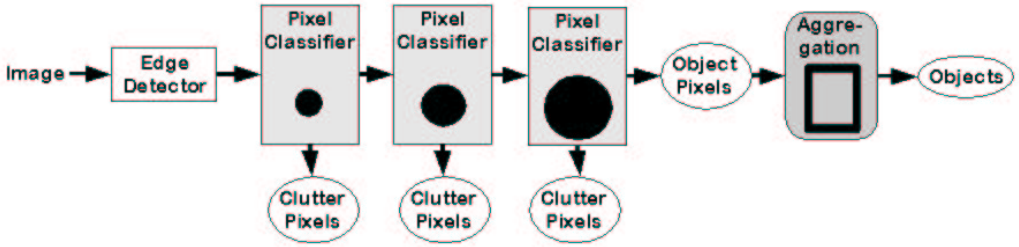
Table 4.2: This table lists the various parameters which control the behavior of cascades trained with aggregation-sensitive part classifiers.

In summary, our strategy for optimizing part classification and aggregation in a single procedure amounts to growing a decision tree which attempts to discriminate between pixels in the training images, and pruning the tree to discriminate between boxes of pixels in the holdout images.

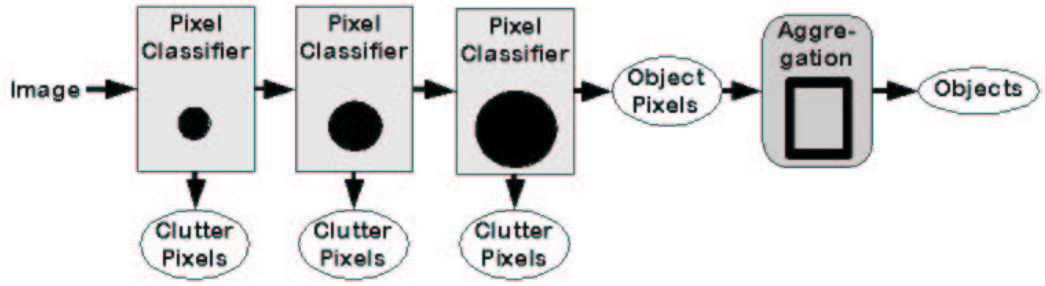
Figure 4.6 shows block diagrams that compare the structure of the classifier cascades as described in Chapter 3, Section 4.1, and Section 4.2. Also, Table 4.2 summarizes the parameters relating to part classification and aggregation that must be manually set or automatically estimated during training to use aggregation-sensitive part classifiers.

4.3 Alternative ROC Techniques

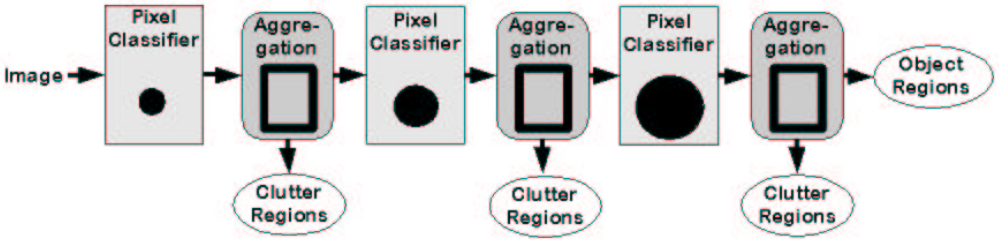
A drawback of our fixed detection rate pruning strategy is illustrated in Figure 4.7. An ROC curve is computed for the classifier scores for the edge pixels in in Figure 3.8(a). Fixed detection rate pruning is based on grading ROC curves and selecting a decision threshold by focusing on a single operating point, $(fp, \theta * np_+)$ and using that point to grade the ROC curve and select a decision threshold. The parameter θ must be set by the user. Consider operating points on the curve $(0.9879, 0.9795)$ (in red) and $(0.7823, 0.9774)$ (in



(a) Default



(b) Edge Operator Probes



(c) Aggregation Sensitive Part Classifier Training

Figure 4.6: Block diagrams comparing the computational organization of the classifier cascades described in Chapter 3 (Figure 4.2), Section 4.1 (Figure 4.2), and Section 4.2 (Figure 4.2).

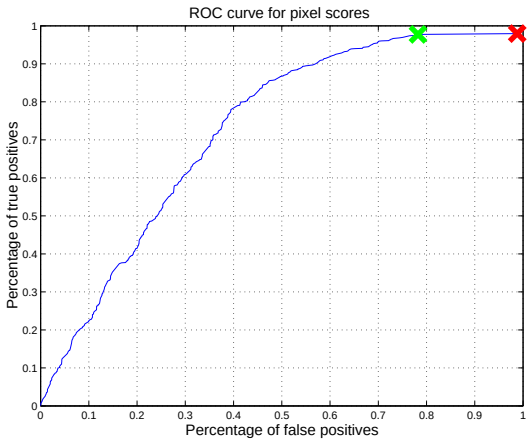


Figure 4.7: An illustration of the inflexibility of using a fixed detection rate criterion for grading an ROC curve and selecting an operating point. The green point, (0.9879,0.9795), has a significantly higher false positive rate than the red point, (0.7823,0.9774), although their true positive rates are very similar. If the true positive threshold is set to the higher of the two, an operating point with significantly higher false positive rate will be selected.

green) corresponding to two very similar thresholds $\theta = 0.9795$ and $\theta = 0.9774$. For us, the difference between the true positive rates for the two operating points is insignificant, but the difference between the false positive rates is substantial. In particular, while the false positive rate for the green operating point (78%) may not be impressive for a stand-alone classifier, removing 22% of the background pixels from images can represent good progress for one classifier in a sequence. Meanwhile, the red operating point represents a classifier that is having practically no impact at all– almost none (1.3%) of the background pixels are discarded by the cascade, and almost none (2%) of the object pixels are discarded either.

The inability to select the green operating point over the red operating point at run time is a limitation of our fixed detection rate scheme. In particular, if the user sets $\theta = 0.9795$, then the red operating point will be selected, even though the green operating point has a very similar true positive rate and a far superior false positive rate. In fact, the CROC plots showing pixel classification results for the cart and chair (Figure 3.12(c) and 3.12(d)) illustrate the opposite side of the problem. There, we set $\theta = .98$, meaning that 2% of the object pixels were discarded by the first classifier in the cascade, and of the 98% remaining, 2% were discarded by the second classifier in the cascade, and so on. This is why, for successive classifiers in the cascade (moving from right to left on the CROC plot), the true positive rate dips from θ to $\theta * \theta$ to θ^3 and so on. The point is that even if each of the ROC curves for those classifiers contained operating points $(fp + \epsilon, 1)$ corresponding to very similar false positive rates $fp + \epsilon$, but a perfect true positive rate of 100%, those superior operating points would not be selected by the fixed detection rate threshold function.

We overcome the brittleness of fixed-detection-rate pruning by considering alternative approaches for ROC grading and threshold determination based on a model of misclassification risk. In this model, each misclassification error incurs a constant numerical cost, and our goal is to find the decision tree which minimizes the sum of all costs on the training set.

Specifically, the cost incurred by each false negative is c_{fn}, and the cost of each false positive is c_{fp}. These penalties encode how important the two types of mistakes are in the overall context of the recognition algorithm. For example, consider a recognition scenario in which a part classifier classifies edge pixels in the scene, and passes the results to an aggregation algorithm which is known to be highly robust to false positive edge pixels, but whose performance decays dramatically if more than a few edge pixels belonging to the target object have been accidentally discarded by the part classifier. In this case, the false positive cost c_{fp} would be relatively low, and c_{fn} would be relatively high. In this example, the costs can either be set by hand or estimated using training data. For instance, a large set of simulated part classification results can be generated corresponding to a range of false positive and false negative rates, and each of these simulated data sets are then given as input to the aggregation step. The results are evaluated to determine how aggregation performance degrades with different levels of false positives and false negatives.

Here we explore two techniques for incorporating misclassification costs into pruning. The first, called *cost-based ROC analysis*, assumes that good estimates of c_{fp} and c_{fn} are known *a priori* [14] [113] [27]. In other words, cost-based ROC analysis assumes that the impact of individual misclassification errors on overall system performance is known at training time. The second technique, called *area-under-the-ROC-based* (or *AUROC-based*) analysis [90] [15], assumes that the real, underlying c_{fp} and c_{fn} are unknown at training time and optimizes the decision tree to produce low total costs for all possible values of c_{fp} and c_{fn}.

In more detail, cost-based ROC analysis sets the decision threshold so that it corresponds to the operating point (fp, tp) which minimizes the sum of costs $c_{tot} = c_{fp} * fp + c_{fn} * fn$. The sum of costs c_{tot} is the ROC grading criterion. In the Figure 4.7 example given earlier, the green operating point has a lower sum of costs than the red operating point for any setting of costs such that $c_{fn} \geq c_{fp} * .0102$. A different sort of cost-based pruning was presented in a paper by Bradford *et al* [14]. There, each leaf of the tree contains a binary classification rather than a continuous score, so that when the holdout set is classified by the tree, each leaf l contains some number fp_l of false positives and some number fn_l of false negatives. Subtrees are replaced with a leaf $l2$ if the sum of costs $fn_{l2} + fp_{l2}$ is

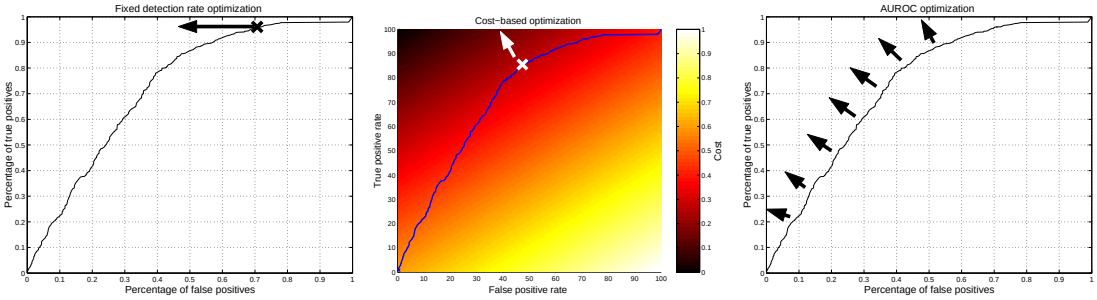
less than the sum of all local costs $fn_l + fp_l$ for all leaves l in the subtree. In other words, pruning is based on a criterion that is local to the subtree being pruned, while in our case the subtree is pruned based a global measure of the performance of the entire decision tree.

AUROC-based pruning, on the other hand, uses the area under the ROC curve as a grading criterion during pruning, following Bradley [15] and Provost *et al.* [90] who used the area under the curve to assign an overall quality measure to a variety of classifiers. The area under the curve provides a measure of the sum of all misclassification costs for all possible settings of c_{fp} and c_{fn}. While AUROC provides an ROC grading criterion, it does not imply a straightforward approach for selecting operating points, as the fixed detection rate and cost-based methods do. In our experiments in Chapter 5, when we use AUROC to grade ROC curves, we employ a cost-based threshold function.

In summary, the three ROC grading criteria we consider can be written mathematically by representing the ROC curve as a function r which maps false positive rates to their corresponding true positive rates along the curve. The three criteria are:

- **Fixed detection rate**. $grade(r) = fp \mid r(fp) = \theta * tp$
- **Cost-based**. $grade(r) = \min_{fp} c_{fp} * fp + c_{fn} * (np_+ - r(fp))$
- **AUROC-based**. $grade(r) = \int_{fp} r(fp)$

Additionally, since our pruning procedure consists of greedily increasing these criteria by discarding subtrees, we can visualize what kind of effect the pruning procedure is trying to make on the ROC curve, given the different grading criteria. Illustrations of these effects are shown in Figure 4.8. When the grading criterion is based on a fixed detection rate (Figure 4.8(a)), pruning isolates the operating point for a particular true positive rate (shown "X") and tries to push that point toward a lower false positive rate. For an AUROC-based grading criterion (Figure 4.8(c)), we are trying to increase the area under the ROC curve. For cost-based grading (Figure 4.8(b)), we impose a cost function $c_{tot}(fp, tp) = c_{fp} * fp + (np_+ - tp) * fn$ over the plane of all possible false positive rates and true positive rates. The point along the curve which minimizes the cost (shown "X") is moved toward lower cost.



(a) Fixed Detection Rate (b) Cost-based (c) AUROC

Figure 4.8: Schematic of how the ROC criteria improve an ROC curve. Figure 4.8(a): Fixed detection rate optimization focuses on moving the operating point for a particular detection rate (here 96%) toward lower false positives. Figure 4.8(b): Cost-based optimization imposes a cost function on the plane of all possible false positive and true positive rates based on c_p and c_{fn} (here, .4 and .6). Points with low costs are in black, high costs are in white. The point along the curve which minimizes this function, here (0.4736,0.8552), is moved toward lower cost. Figure 4.8(c): AUROC-based optimization tries to increase the area under the curve.

Chapter 5

Experiments

In this chapter, we present a set of experiments which explore the performance of part classifiers and aggregation schemes based on the baseline algorithm presented in Chapter 3 and the extensions presented in Chapter 4. We test the extensions on the "ladder in the lab" image set shown in the top row of Figure 3.13 and present the results as cascade ROC curves for pixel-level detection rates and box-level detection rates. Additionally, we present similar results on other image sets containing a stool.

These experiments illustrate a number of advantages of the extensions in Chapter 4. In Section 5.3 we show that edge operator probes can enable bottom-up, shape-based localization in images with weak edges. Based on an initial family of edge operators, we train classifier cascades which are able to discard most pixels among the background in novel images while retaining a high percentage of pixels on the object. In Section 5.4 we present results that show that aggregation-sensitive part classifier training makes box-level summary of classified pixels more reliable than if aggregation and part classification were trained as two separate modules. Furthermore, Section 5.6 summarizes an empirical comparison of the three ROC optimization schemes presented in Section 4.3, which conforms to our intuition that AUROC-based and cost-based optimization can be more flexible and converge to favorable operating points quickly.

In Sections 5.3 and 5.4 we observe that the performance of cascades of edge operator probes, with or without aggregation-sensitive training, degrade when the test images contain dramatically new backgrounds, new object scales, or occlusions not present in the training data. This follows our intuition that the cascades are tuned to specific objects and backgrounds. Section 5.3 also shows that training pixel classification cascades and aggre-

gators separately can lead to unpredictable aggregation performance. Section 5.8 describes why aggregation-sensitive training is difficult to apply to image sets in which the object takes up a large percentage of the image. Section 5.7 explains that the performance of cost-based classifier cascades can be sensitive to parameters related to misclassification costs and cascade construction.

We also present experiments that point out two overall characteristics of our algorithms. First, Section 5.5 shows that the training data class distribution– that is, the relative percentage of object and background examples in the tree-growing and holdout sets– can have a major impact on classification performance. Then, Section 5.9 presents results which suggest that the performance of edge probe cascades can be compromised by overly-aggressive feature selection.

First, Sections 5.1 and 5.2 describe the image sets and evaluation criteria used in these experiments.

5.1 Data

As can be seen in Table 3.13, bottom row, the lab images were the most challenging for the baseline algorithm in terms of false positive pixels; a cascade of edge probes was able to remove about 80% of all background edge pixels from the lab images, compared to roughly 90% in the images of the ladder in other environments. For this reason, we chose to evaluate variants to our baseline algorithm using this set of images. To review, we took 122 lab images by placing the ladder in front of a cluttered lab space and taking images covering a range of ladder poses, camera poses, and clutter arrangements. The camera translated from one side of the space to the other between views; the elevation of the camera varied between 1.6m and 1.75m; and once every five views the ladder was rotated in place and the clutter objects were shuffled. More example images from the lab data set are shown in Figure 5.1.

We partitioned the image set into a tree-growing set of 55 images, a holdout set of 55 images, and a test set of 12 images. We trained cascades of edge probes and aggregation filters using the tree-growing and holdout sets; results in terms of pixel-level and box-level detection rates for the cascade phases are reported as cascade ROC curves.

For a second set of images, we took a set of 20 videos of a barstool at various viewpoints (Figure 5.2). As in the ladder images, the clutter objects shuffle between video clips, and

Figure 5.1: Example images of the ladder drawn from the lab image set.

the stool rotates on the floor, but the scale of the object does not vary more than about 15% across all frames. We used a 2-frame optical flow tracker to label the position of the stool in each frame of the videos. The videos are partitioned into five categories: the first is a set of 7 video sequences containing images of the stool in front of minimal internal clutter[1](Figure 5.2(a) and 5.2(b)); the second set of 4 videos show the object in front of more challenging interior clutter (Figure 5.2(c)); in the third image set, the stool is slightly occluded (Figure 5.2(d)); in the fourth, the stool appears at a scale that differs from the scale range seen in training images (Figure 5.2(e)); in the fifth, the stool is in another part of the room (Figure 5.2(f)); and in the sixth, the stool is in a different set of rooms altogether (Figures 5.2(g) and 5.2(h)). Call these image sets the "low-internal-clutter," "high-internal-clutter," "occlusion," "scale," "other side," and "other room" images respectively. We used these images as a motivation for using edge operator probes rather than edge probes as image features, since some of the edges on the target object are quite weak. Here, we employ two training sets of varying complexity to understand how our techniques respond to variations in clutter characteristics between training and test images. In particular, the first training set consists of a subsest of the low-internal-clutter videos, and the second is a collection of low-internal-clutter and high-internal-clutter images.

5.2 Evaluation

For each experiment, we extract features from the training images and use them to optimize part classifiers and aggregation filters. When we test the classifiers and aggregation steps on test images, we evaluate the performance of each phase of the cascade in terms of part classification and aggregation. More specifically, we pass each classifier in the part

[1]"Internal clutter" refers to clutter which appears through holes in the object, for example between the legs and leg supports of the stool.

(a) Low Interior Clutter

(b) Low Interior Clutter

(c) Higher Interior Clutter

(d) Occlusion

(e) Scale Variation

(f) Different Part Of The Room

(g) Other Room 1

(h) Other Room 2

Figure 5.2: Example images of the stool. Figure 5.2(a) shows a typical image in the training set. The rest are images from various test sets: minimal interior clutter (Figure 5.2(b)), higher interior clutter (Figure 5.2(c)), occlusion (Figure 5.2(d)), different scale (Figure 5.2(e)), different part of the same room (Figure 5.2(f)), and different rooms (Figure 5.2(g) and 5.2(h)).

(a) (b)

Figure 5.3: Example image of the stool with its associated labeling. Figure 5.3(a) shows the input image, and white pixels in Figure 5.3(b) are labelled as belonging to the target object.

classifier cascade over the image in sequence. For the kth cascade phase, we evaluate the percentage of true positive pixels versus percent of false positive pixels for a classifier cascade consisting of phases 1 through k. Given those true positive and false positive pixels, we scan an aggregation filter over the image in order to evaluate how well an aggregation filter would perform if the part classifier cascade only consisted of the first k phases. Aggregation performance is plotted in terms of percentage of true positive boxes– that is, the percentage of bounding boxes surrounding the target object that were identified as an instance of the target object– and number of false positive boxes.

For each test image, we evaluate the number of true positive and false positive pixels by referring to a version of the image whose pixels have been hand-labeled as belonging to the object or background. However, evaluating the rates of true positive and false positive boxes is slightly less straightforward. In particular, if the region of the image in which an instance of the target object is detected overlaps the true target object region, but only partially, it is unclear whether to count it as a true positive box or a false positive box. In these experiments, we count a bounding box as a true positive if it overlaps with the true bounding box of the target object by 75% or more.

5.3 Edge Operator Probes

In this experiment we tested the use of edge operator probes in training a pixel-level classifier cascade. Aggregation-sensitive training was *not* employed, i.e. we trained the cascade of classifiers to classify individual image pixels without using the modification in Section 4.2. The results of these experiments show that on novel test images taken in similar

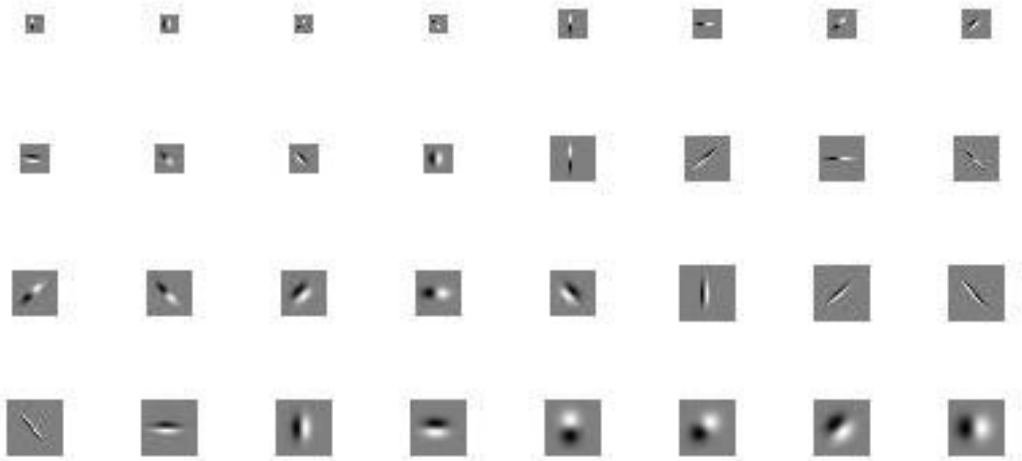
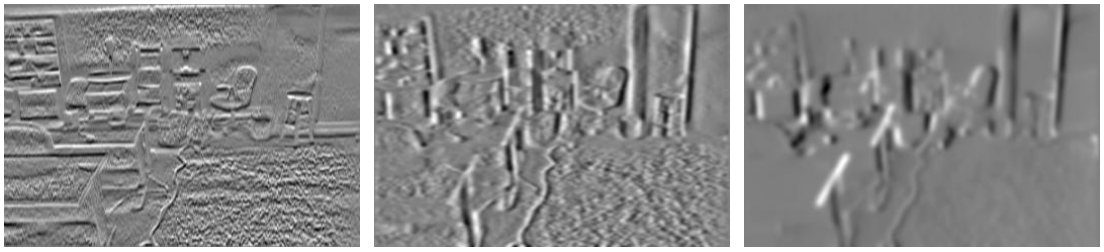


Figure 5.4: The set of 32 edge operators selected using the reconstruction-based algorithm in Section 4.1.1 for image features on the stool images.

environments to the training images, cascades of edge operator probes are effective at identifying a high percentage of object pixels and discarding a high percentage of background pixels. Also, our results show that applying an aggregation filter to an image classified by a cascade of edge operator probes can result in high box-level localization performance if enough training data is available, but when training data is limited, the aggregation performance is unpredictable. Also, our results indicate that the pixel classification performance of a cascade of edge operator probes decays gracefully as the test images diverge farther from the training images.

We examine pixel-level classification performance across changes in clutter characteristics in the test images; in particular, we train a pixel-level classifier cascade on the stool images with low internal clutter (Figure 5.2(a)) and evaluate the classification performance on the different categories of test images. A total of 266 training images were gathered by selecting every fifth frame from a set of four videos of the stool in front of minimal internal clutter. For test images, we gathered every fifth frame of all the videos in each of the 8 stool image categories. The number of test images in these categories are indicated in Figure 5.6(a). Additionally, we trained another cascade on both low-internal-clutter and higher-internal-clutter images (Figure 5.2(a) and 5.2(c)). In this case, there were 466 training images, gathered by selecting every tenth frame from a collection of four low-internal-clutter videos and three high-internal-clutter videos. The number of test images in each image category for this cascade is listed in Figure 5.6(b).



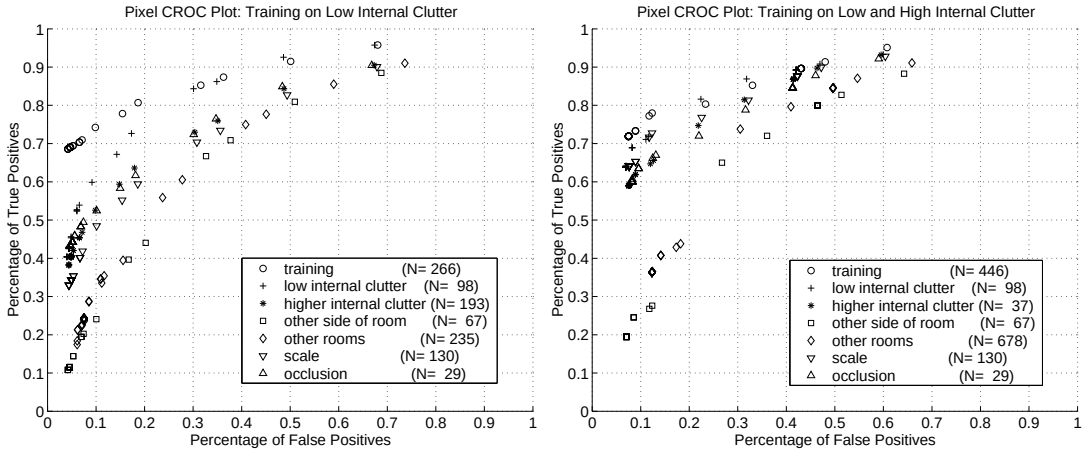
(a) (b) (c)

Figure 5.5: Examples of how some of the selected edge operators respond to the input image in Figure 5.3(a). Figures 5.5(a), 5.5(b), and 5.5(c) show the responses of the operators in (row,column) (1,5), (3,4), (4,8) of Figure 5.4.

We selected features using the techniques in Section 4.1.1. In particular, we began with a set of 640 edge operators corresponding to all possible combinations of the following parameter values: $\sigma_x = \{2.1, 3.3, 5.1, 6.3, 8.1, 9.3, 11.1, 12.30\}$, $\sigma_y = \sigma_x * \{.1, .32, .55, .77, 1.0\}$, $\phi = \{0, .8, 1.6, 2.39, 3.19, 3.99, 4.79, 5.59\}$, and derivative with respect to x and y. We removed edge operators using Algorithm 4, setting the correlation threshold h to .8, and randomly selecting a set of 50 images from the low-internal-clutter training set (Figure 5.2(a)) for the representative images $\mathcal{T}$. The resulting set of 32 edge operators were used during training to form operator-offset pairs (See Figure 5.4 and Figure 5.5). When training each cascade phase, we applied the filter method described in Section 4 to reduce the full set of operator-offset pairs to the 5 which maximized the discrimination criterion. Misclassification costs were set to $(c_{fp}, c_{fn}) = (.01, .99)$.

Pixel classification results for the two cascades are plotted and summarized in Figure 5.6, and an anecdotal result on a typical image for the low- and high-internal-clutter cascade is shown in Figure 5.7. Also, pixel and object true and false positive rates after the final cascade phase are summarized in Table 5.1. Note that at as more phases are added to the cascade, more and more background pixels are discarded, and a high percentage of pixels on the stool remain in the image. After pixel classification, it is possible to post-process the result to remove isolated false positives, for example by median filtering the image (Figure 5.7, bottom right).

Looking at each graph in Figure 5.6 individually, we see that for each type of test image, the cascades are able to filter out 90% to 95% of all false positive pixels from the images, and about 50-70% of object pixels in images similar to the training images (the +marks in Figure 5.6(a), +and *in Figure 5.6(b)) are retained. True positive rates decay as the divergence between test images and training images increases; for example, vary between the image types: for example, in both cases the true positive rates for images taken in other rooms ($\diamond$) and on the other side of the room ($\square$) are lower than true positive rates for images



(a) Training on Low-Internal-Clutter Images

(b) Training on Low- and High-Internal-Clutter Images

Image Type	Low Int. Clutter		High Int. Clutter	
	TP	FP	TP	FP
Training	0.686	0.042	0.719	0.075
Low Int. Clutter	0.404	0.040	0.638	0.069
High Int. Clutter	0.382	0.044	0.590	0.075
Other Side Of Room	0.108	0.042	0.194	0.071
Other Rooms	0.184	0.061	0.363	0.123
Scale	0.330	0.044	0.641	0.076
Occlusion	0.434	0.047	0.600	0.082

(c) Summary

Figure 5.6: Pixel CROC plots for classifier cascades trained using edge operator probes and cost-based optimization. Each point corresponds to a pixel detection operating point for the sets of test images. Figure 5.6(a) show results for a cascade trained strictly on low-internal-clutter images, and Figure 5.6(b) shows results for a cascade trained on low- and high-internal clutter images. The number of images in the training and test sets is indicated in the legend. Figure 5.6(c) summarizes the pixel true positive and false positive rates achieved by the low-internal-clutter and high-internal-clutter cascades on the various image types.

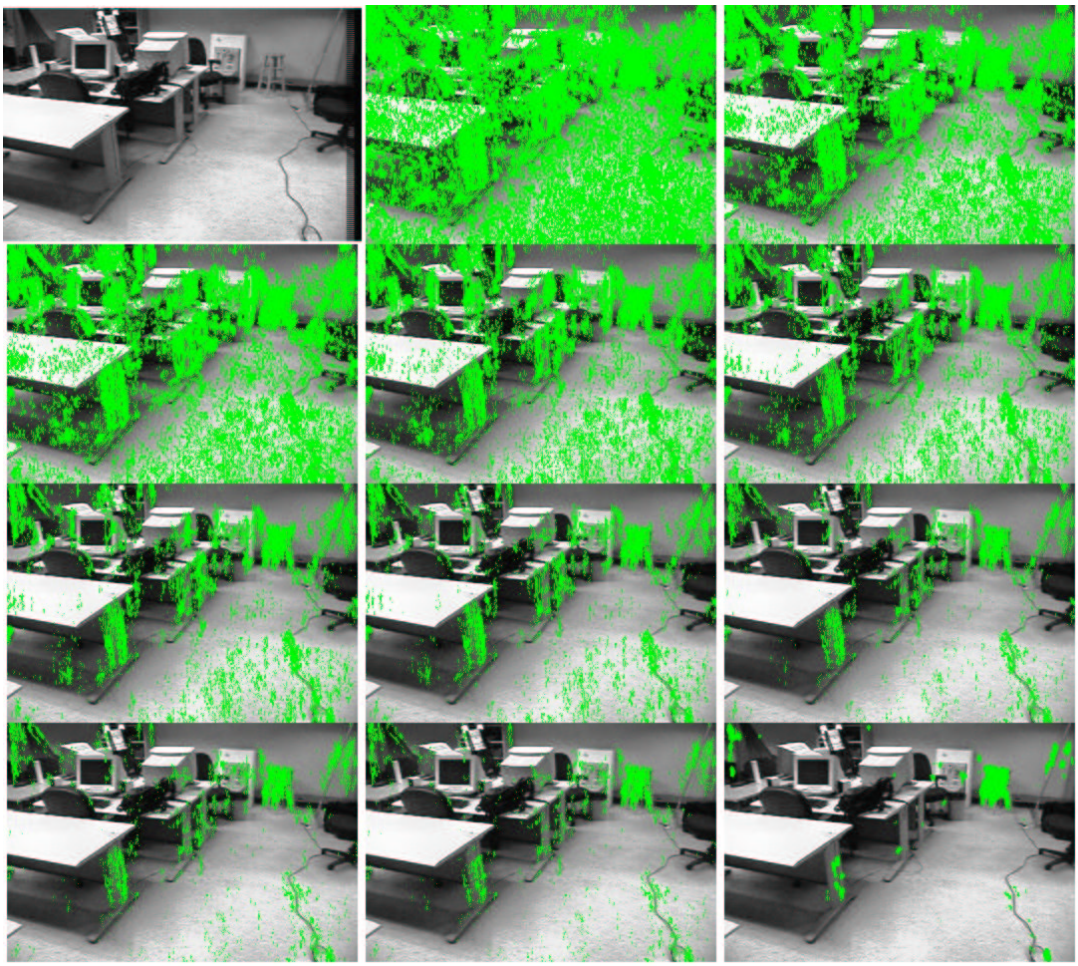


Figure 5.7: Example of pixel classification results at many cascade phases. The input image is shown at top left; the next ten images (in left-to-right, top-to-bottom sequence) show pixel classification results for the pixel classification cascade described in Section 5.3. Green pixels are classified as "object." Shown are the results after the following cascade phases: 1,2,8,9,10,11,12,13,17,26. The last photo in the sequence (bottom right) shows a median-filtered version of the final pixel result (bottom center).

with varying scale (∇) or occlusion ($\triangle$). The true positive rates for the scaled and occluded images are in turn the same as, or lower than, the images with little scale changes with low internal clutter (+) or higher internal clutter (*). This conforms to our intuition that since our cascades are discriminative, they are tuned to the specific objects and backgrounds they are trained on, and are not designed specifically to generalize to new conditions.

Between the two graphs, we see the predictable result that for the cascade trained on high- and low-internal-clutter images (Figure 5.6(b)) the true positive rate on high-internal-clutter images (*) does not decay with respect to the true positive rate on low-internal-clutter images (+) the way it does in the low-internal-clutter cascade. This is to be expected because the high-internal-clutter images are present in the training images. Interestingly, however, true positive rates for all types of images increases significantly in the low-and-high-internal-clutter cascade; for example, about 60% of the object pixels in the less challenging images (+,*,∇,$\triangle$) are retained, versus about 40% in the low-internal-clutter case. This is likely due to the fact that almost twice as many training images (446 *vs.* 266) were used to train the cascade depicted in Figure 5.6(b).

Box-level classification rates are shown in Figure 5.8 and 5.9. To review, at each cascade phase we first screen each test image through the pixel classifier at that phase, arriving at a binary classification for each pixel. We then scan the classified image with an aggregation filter that returns the number of positively-classified pixels inside a bounding box. By thresholding this aggregation score, we arrive at box-level localizations of object instances in the test image. We arrive at a threshold for the sum of pixel scores using aggregation scores from the training images; specifically, we scan the training images with the aggregation filter and record the aggregation scores for box locations which cover the target object, and box locations covering the background. We select the aggregation score threshold using a fixed detection rate criterion: that is, we select the aggregation score threshold so that a very high percentage of true positive boxes in the training images (here, 98%) pass the threshold. Each plotted point shows an operating point for a box-level classifier applied to the classified image after some number of pixel-level classification cascade phases. For example, the point marked "1" at (9.68,0.83) represents the results of applying the aggregation filter to the results of applying a single-phase pixel classification cascade to high-internal-clutter images. The point marked "2" at (7.69,0.86) shows the aggregation filter performance when applied to the same images classified by a two-phase pixel classification cascade.

Consider the results on the low-internal-clutter cascade (Figure 5.8). The most impor-

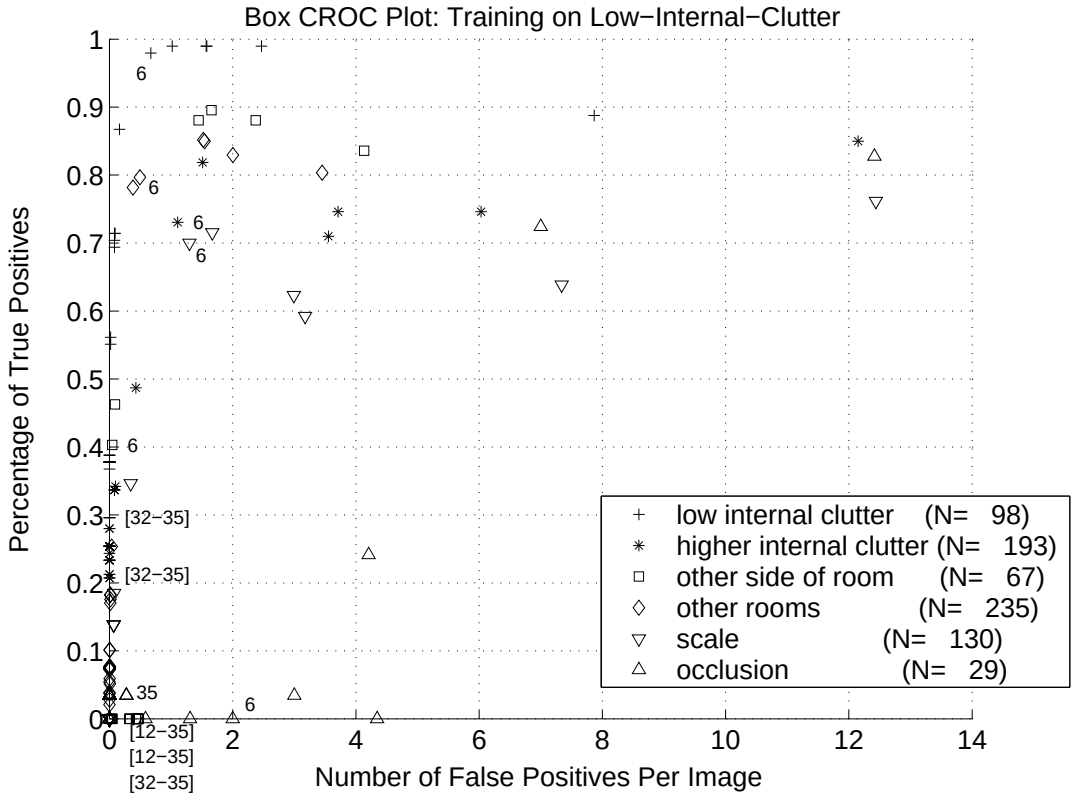


Image Type	Phase 6		Phase 35	
	TP	FP	TP	FP
Low Int. Clutter	0.980	0.673	0.296	0
High Int. Clutter	0.731	1.109	0.207	0
Other Side Of Room	0.403	0.045	0	0.328
Other Rooms	0.782	0.382	0.021	0.001
Scale	0.7	1.3	0	0
Occlusion	0	2	0	0

Figure 5.8: Box CROC plots for classifier cascades trained using edge operator probes and cost-based optimization. Each point corresponds to a box detection operating point for the sets of test images. Which phase of the cascade the point corresponds to is indicated by a number below or beside it. A range of numbers in brackets indicates that the cascade maintained the same number of false positive boxes and true positive boxes over the range of phases indicated. The number of images in the training and test sets is indicated in the legend. The table lists the final true positive and false positive rates for the various image types.

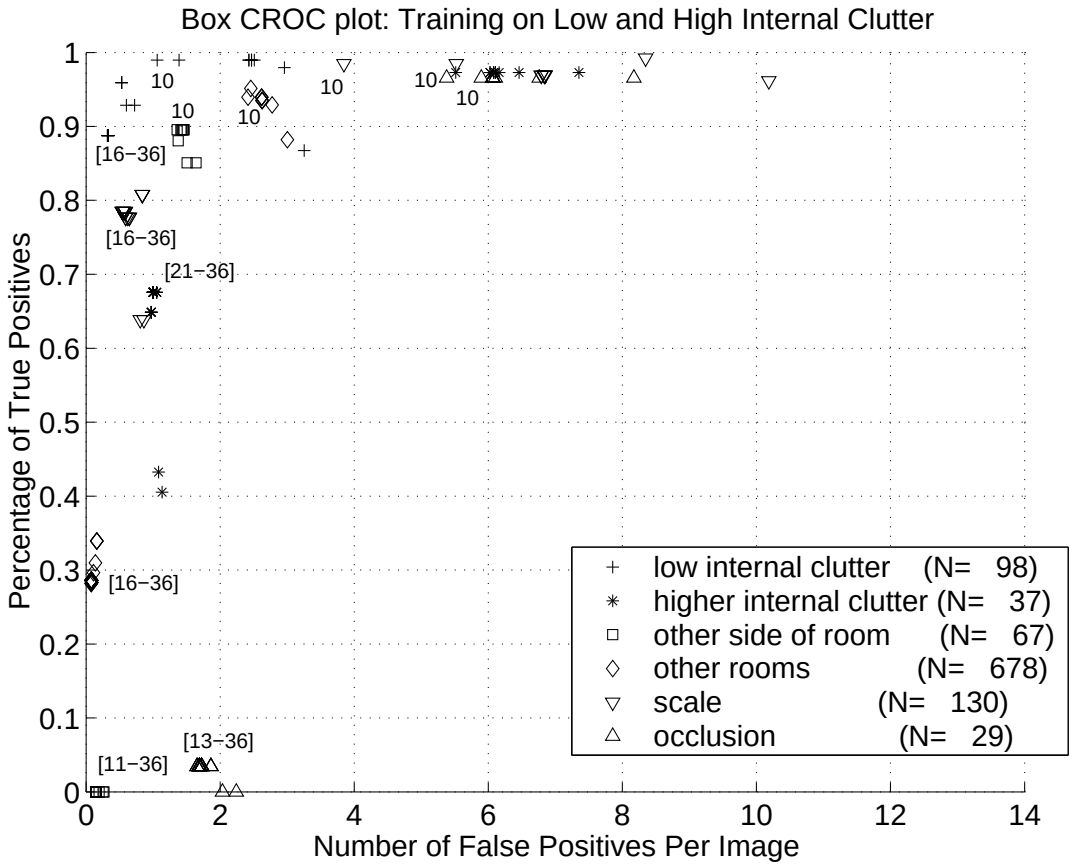
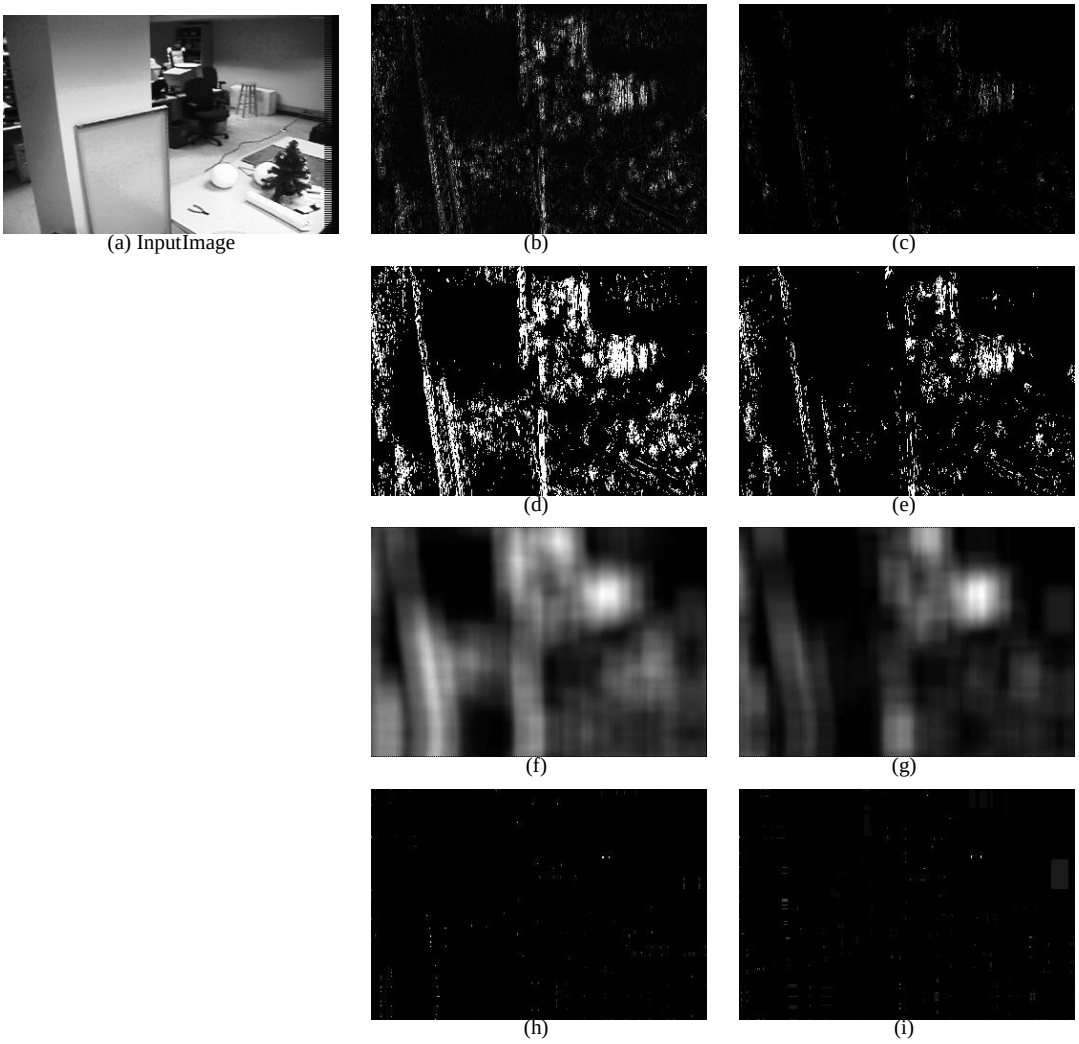


Image Type	Phase 10		Phase 36	
	TP	FP	TP	FP
Low Int. Clutter	0.990	1.061	0.888	0.327
High Int. Clutter	0.973	5.514	0.676	1
Other Side Of Room	0.896	1.358	0	0.149
Other Rooms	0.951	2.459	0.286	0.078
Scale	0.985	3.846	0.785	0.546
Occlusion	0.966	5.379	0.034	1.724

Figure 5.9: Box CROC plots for classifier cascades trained using edge operator probes and cost-based optimization. Each point corresponds to a box detection operating point for the sets of test images. Which phase of the cascade the point corresponds to is indicated by a number below or beside it. A range of numbers in brackets indicates that the cascade maintained the same number of false positive boxes and true positive boxes over the range of phases indicated. The number of images in the training and test sets is indicated in the legend. True positive and false positive rates after phases 10 and 36 are summarized in the table.



(a) InputImage (b) (c) (d) (e) (f) (g) (h) (i)

Figure 5.10: An anecdotal result of classifying the image in Figure 5.10(a) using a cascade of edge operator probes *without* aggregation-sensitive part classification. The left column shows classification results after a single cascade phase, and the right column shows the results after the 8th and final cascade phase. First row: Each pixel is assigned a pixel classifier score. Second row: We threshold that score to determine which pixels are discarded and which move on to the next cascade phase. Third row: Given a set of pixels which have passed the threshold, we scan the image with an aggregation filter which adds up the number of pixels inside an object-sized box. Here, each pixel represents the aggregation score of a bounding box whose upper lefthand corner is anchored at the pixel. Fourth row: The result of applying non-maximum suppression to the images in the third row.

tant point to note is that at first, adding pixel classification phases to the cascade helps box classification performance, but in many cases too many pixel classification phases can actually hurt box classification performance. As a concrete example, compare the box classification performance of aggregation filters which follow two different pixel classifier cascades: the first contains only six phases, and the other contains the full set of 35 phases. The first aggregation filter attains good classification results on many of the image types (see the points in Figure 5.8 marked "6"): for example, true positive rates of 98%, 72%, 70%, and 79% on low-internal clutter, high-internal-clutter, scale, and other room image sets respectively. The second aggregation filter (see the points marked "35") does much worse, missing most true positives in most images. The reason for the performance dropoff is that the threshold for accepting an aggregation score in a test image as an object instance is set by the aggregation scores on training images; that is, we count the number of true positive pixels inside boxes covering the object in training images, and set the aggregation score threshold so that it is just below a high percentage of those scores. Since the number of true positive pixels in the training images is dramatically higher than the number of true positive pixels in the test images at later cascade phases (see Figure 5.6(a)), the aggregation score threshold will be unrealistically high. However, these results illustrate two points. First, that aggregation filters can effectively localize the object after only a few cascade phases, and second, that the performance of the aggregation filter can be poor if the aggregation threshold is unrealistically high.

Turning to the low-and-high-internal-clutter cascade (Figure 5.6(b)), we see that box classification performance improves dramatically. In particular, after the full set of 36 cascade phases, the low-internal-clutter images (+) achieve a true positive rate near 90% with about .5 false positive boxes per image. Performance on the higher-internal-clutter images (*) and scale images (∇) for the full cascade is lower but still reasonable. For the images of occlusion ($\triangle$), other side of the room ($\square$), and other rooms ($\diamond$), we see the same effect as in Figure 5.6(a): for some of the early cascade phases, box classification is reasonably effective, and in later phases the aggregation score threshold is unreasonably high.

Overall, this experiment indicates that cascades of edge operator probes can effectively discriminate between object pixels and background pixels in real images. Furthermore, joining these cascades with aggregation filters must be done with care for good performance.

5.4 Edge Operator Probes And Aggregation-Sensitive Part Classifier Training

This experiment addresses the use of edge operator probes (Section 4.1), together with aggregation-sensitive part classifier training (Section 4.2. We show that effective object-level localization can be achieved by coupling the optimization of part classifiers with aggregation filters. Our results show that applying a pixel-level classifier and aggregation filter at each cascade phase can effectively eliminate large portions of the image from consideration in a single step. Our results also indicate that the performance of cascades using these extensions degrade when novel occlusions and changes in scale are present at run time, and that while the cascades correctly identify the portion of the image containing the object, care must be taken to not trigger multiple, overlapping object instances in that region.

At each cascade phase we first use the feature selection strategies from Section 4.1.1 to select a set of operator-offset pairs, and use edge operator probes corresponding to those operator-offset pairs as image features. We use the images in the tree-growing set to build a decision tree to discriminate object pixels from background pixels based on the edge operator probe values. Then, we compute classifier scores for each pixel in the holdout images and compute aggregation filter scores for each background region and foreground region of those images by summing the pixel classification scores. Subtrees are removed from the decision tree if doing so improves an ROC grading criterion on the ROC curve of aggregation filter scores. Once the tree has been pruned, we discard all pixels from the training images if they are not inside any retained bounding boxes. At run time, we scan the image with the pixel classifier and aggregation filter for each cascade phase in turn.

The image features and feature selection process were the same as for the experiment in Section 5.3. That is, we used the 32 edge operators shown in Figure 5.4, the same filter method for selecting features at each cascade phase, and the same number (5) of operator-offset pairs per cascade phase.

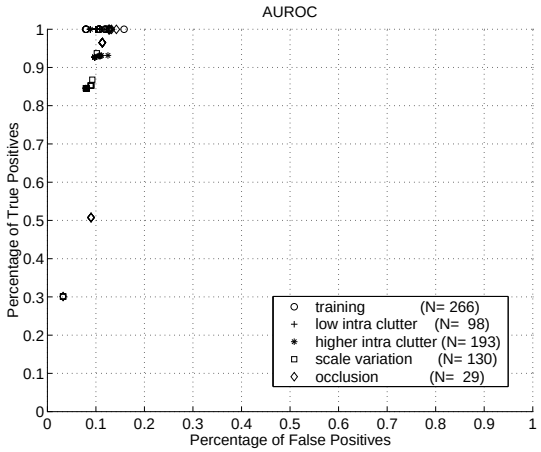
We trained three different classifier cascades on the low-internal-clutter image set. Each cascade used a different ROC optimization scheme to train decision trees. For the cascade based on the fixed detection rate criterion, our target true positive rate for each classifier in the cascade is .98, and if a classifier fails to reduces the number of false positive object instances over all training images by at least 1, we discard that classifier and train a classifier using image features from the the next aperture size. We employed misclassification costs

$(c_{fp}, c_{fn}) = (.01, .99)$ to optimize a cascade which used the cost-based criterion. For the cascade based on optimizing the AUROC criterion, we selected operating points on ROC curves by finding the point on the curve which minimizes the sum of costs (Eq. 4.3), with the same misclassification costs as in the cost-based optimization cascade. The only major difference between the setup of this experiment and that of Section 5.3 is that the latter trained pixel classification cascades first and applied aggregation filters *post facto*, while here we train pixel classifiers and aggregation filters together.

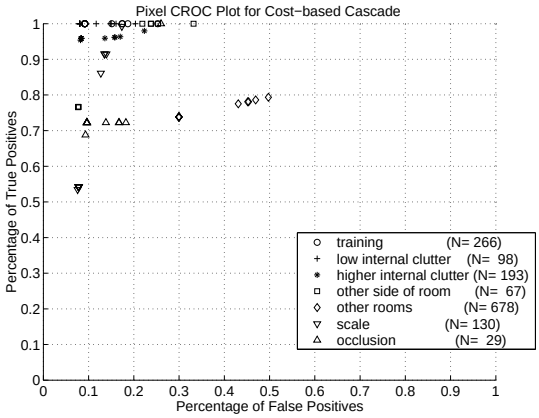
Pixel classification results are shown for the three cascades in Figure 5.11, and box classification rates are shown in Figure 5.12. Also, pixel and box classification rates are summarized in Table 5.1 and compared with similar cascades trained without aggregation sensitivity. For each cascade, for each image set, the number of false positive pixels among the background is reduced to about 10%, and for most images the percentage of true positive pixels retained by the cascade is high, between 80% and 100%. Furthermore, each type of cascade is able to correctly identify the stool in most of the test images with lower internal clutter and higher internal clutter– in these cases, roughly 80% to 100% of the stools are correctly identified, with about 2-3 false positive detections per image. The differences between the three cascades will be discussed in Section 5.6.

The false positive rate in the box classification results is pessimistically high; a typical example of this illustrated in Figure 5.14(i). Clearly, the boxes identified as instances of the stool are located in the correct portion of the image; however, there are two very faint local peaks in the aggregation score in that region, so two instances of the object (which overlap each other significantly) are returned by the algorithm, triggering a false alarm. In fact, many of the false alarms triggered in the experiments in this section are of this type– the aggregation score is only significantly high in the region of the image containing the object, but because of minor variations in the score they produce multiple peaks which are picked up by non-maximum suppression. This problem would be alleviated be smarter post-processing of the aggregation scores.

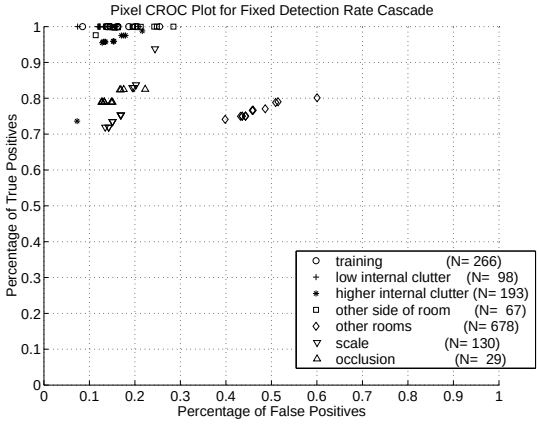
A typical result is shown in Figure 5.13. The images show the results of classification by the cascade after 1,2,4,8, and 14 cascade phases; green regions of the image are still under consideration by the cascade, and all other parts of the image have been discarded. Note that even after a single cascade phase, large sections of the image have been discarded. As more steps in the classifier cascade are applied, more false positive boxes are removed from consideration. Compared to the result shown in Figure 5.10, we see that the cascade phases here remove false positive pixels in contiguous chunks rather than at scattered image



(a) AUROC



(b) Cost Based

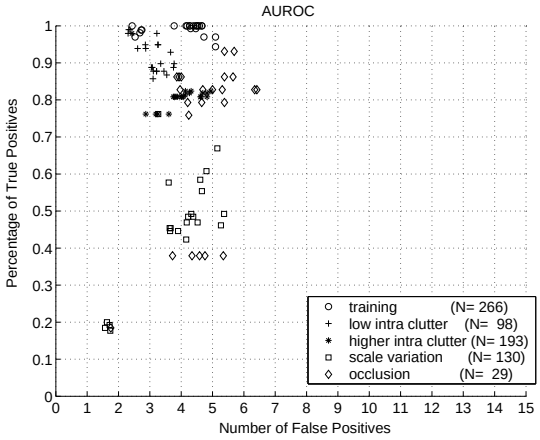


(c) Fixed Detection Rate

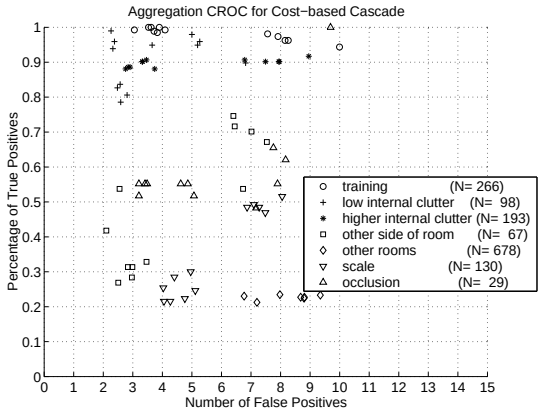
Image Type	Fixed		Cost		AUROC	
	TP	FP	TP	FP	TP	FP
Training	1.000	0.080	1.000	0.091	1.000	0.085
Low Internal Clutter	1.000	0.089	1.000	0.080	1.000	0.074
Higher Internal Clutter	0.846	0.079	0.954	0.082	0.736	0.073
Other Side Of Room	0.301	0.032	0.766	0.077	0.976	0.114
Other Rooms	0.508	0.090	0.738	0.299	0.741	0.398
Scale	*	*	0.534	0.076	0.135	0.719
Occlusion	*	*	0.688	0.093	0.127	0.789

(d) Summary

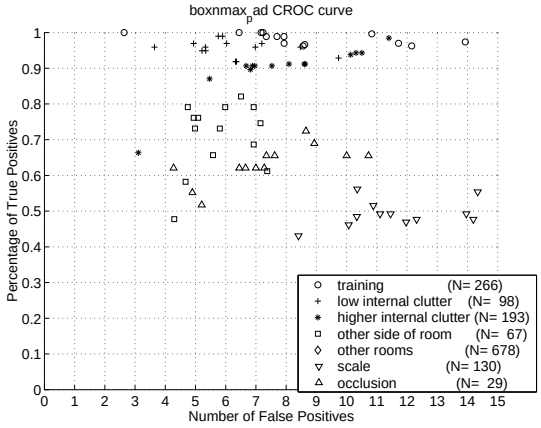
Figure 5.11: Pixel detection CROC curves for classifiers trained using the image features presented in Section 4.1 and aggregation-sensitive training in Section 4.2, trained on the stool images. Figure 5.11(a): Results for classifier cascades which optimized AUROC, cost based, and fixed detection rate criteria are shown in Figures 5.11(a), 5.11(b), and 5.11(c) respectively. Table 5.11(d) summarizes the pixel true positive and false positive rates after the final cascade phase for the three cascade types. The results for the fixed-detection-rate cascade on the scale and occlusion images were lost in a computer failure.



(a) AUROC



(b) Cost Based



(c) Fixed Detection Rate

Image Type	Fixed		Cost		AUROC	
	TP	FP	TP	FP	TP	FP
Training	0.989	2.722	1.000	3.530	1.000	2.639
Low Internal Clutter	0.980	2.429	0.827	2.480	0.959	3.643
Higher Internal Clutter	0.762	3.269	0.881	2.756	0.663	3.109
Other Side Of Room	0.185	1.754	0.284	2.970	0.478	4.299
Other Rooms	0.379	4.586	0.224	8.795	0.226	18.355
Scale	*	*	0.254	4.031	0.431	8.408
Occlusion	*	*	0.552	3.207	0.621	4.276

(d) Summary

Figure 5.12: Aggregation CROC plots for classifiers trained using the image features presented in Section 4.1 and aggregation-sensitive training in Section 4.2, trained on the stool images. Figure 5.12(a): Results for classifier cascades which optimized AUROC, cost based, and fixed detection rate criteria are shown in Figures 5.12(a), 5.12(b), and 5.12(c) respectively. Figure 5.12(d) summarizes the true positive and false positive rates for the three cascades after the final cascade phase. The results for the fixed-detection-rate cascade on the scale and occlusion images were lost in a computer failure.

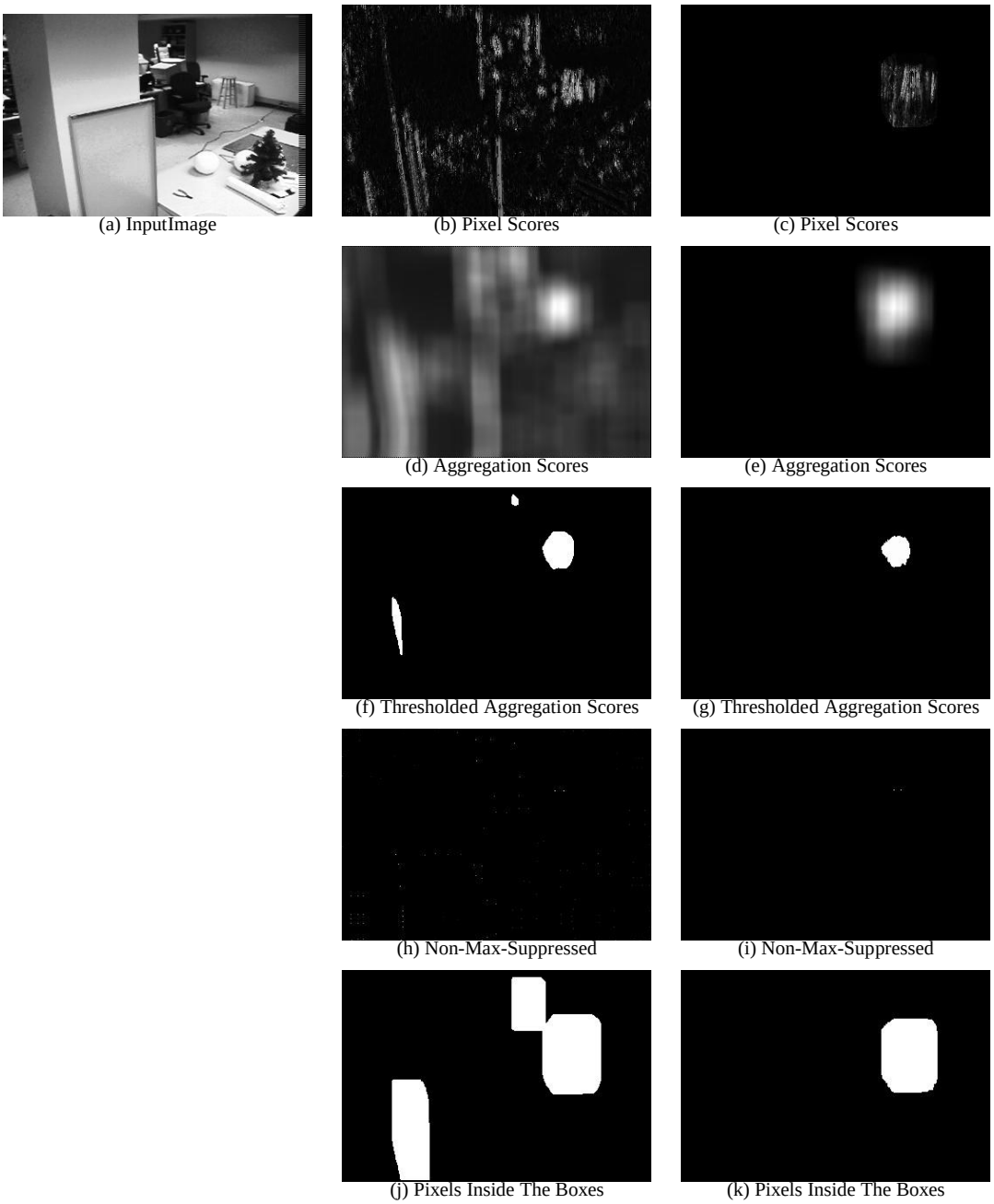
Cascade	Testing On Low Clutter					Testing On Higher Clutter				
	Pixel TP	Pixel FP	Object TP	Object FP	# Test	Pixel TP	Pixel FP	Object TP	Object FP	# Test
EOP, Low Clutter	0.404	0.040	0.296	0	98	0.382	0.044	0.207	0	193
EOP, High Clutter	0.638	0.069	0.888	0.327	98	0.590	0.075	0.676	1	37
EOP, Aggregation-Sensitive	1.000	0.080	0.827	2.480	98	0.954	0.082	0.881	2.756	193
	Testing On Other Side Of Room					Testing On Other Rooms				
EOP, Low Clutter	0.108	0.042	0	.328	67	0.184	0.061	0.021	0.001	678
EOP, High Clutter	0.194	0.071	0	0.149	67	0.363	0.123	0.286	0.078	678
EOP, Aggregation-Sensitive	0.766	0.077	0.284	2.970	67	0.738	0.299	0.224	8.795	678
	Testing On Scale					Testing On Occlusion				
EOP, Low Clutter	0.330	0.044	0	0	130	0.434	0.047	0	0	29
EOP, High Clutter	0.641	0.076	0.785	0.546	130	0.600	0.082	0.034	1.724	29
EOP, Aggregation-Sensitive	0.534	0.076	0.254	4.031	130	0.688	0.093	0.552	3.207	29

Table 5.1: Summary of recognition rates after the final cascade phase for three cascades trained on the stool images using cost-based pruning. The cascade trained using edge operator probes as image features and low-internal-clutter training images is marked "EOP, Low Clutter". The cascade trained with edge operator probe features on low- and high-internal-clutter training images is marked "EOP, High Clutter." The cascade trained with edge operator probes and aggregation-sensitive training on the low-internal clutter images is marked "EOP, Aggregation-Sensitive." Pixel and object true positive/false positive rates after the last cascade phase are listed as Pixel TP/FP and Object TP/FP. # Test gives the number of test images of a specific image type.

locations.

However, detection of the object in all cases is significantly worse for the images containing scale deviations from the training images or occlusions. In particular, the cost-based and AUROC-based cascades only detect the stool in about 20% of the images with scale variation, with 2-5 false positives per image. The fixed detection rate cascade performs somewhat better, finding about 65% of the instances of the stool with 3 false positives per image. Detection performance on the occlusion images is similar. The degradation in performance in these cases is not entirely unexpected since the scale change and occlusion were not present in the training images, and since training of the part classifier is tightly coupled with a fixed-size aggregation filter. In particular, since each image is scanned with one bounding box of a characteristic object size, scale changes in the object mean that this bounding box is more likely to not cover the entire object or contain more clutter pixels.

Figure 5.13: Example pixel classification results for a cascade using edge operator probes, aggregation-sensitive part classification, and fixed detection rate pruning (*i.e.*, Sections 4.1 and 4.2. The input image is shown at top left; the next five images (in left-to-right, top-to-bottom sequence) show pixel classification results for the pixel classification cascade. Green pixels are classified as "object." Shown are the results after the following cascade phases: 1,2,4,8,14.



(a) InputImage (b) Pixel Scores (c) Pixel Scores

(d) Aggregation Scores (e) Aggregation Scores

(f) Thresholded Aggregation Scores (g) Thresholded Aggregation Scores

(h) Non-Max-Suppressed (i) Non-Max-Suppressed

(j) Pixels Inside The Boxes (k) Pixels Inside The Boxes

Figure 5.14: An example of classifying the image in Figure 5.14(a) with edge operator probes and aggregation-sensitive filtering. The left column shows the results of classification after a single cascade phase, and the right column shows results after the 12th and final cascade phase. First row: Each pixel is assigned a pixel-level classification score. Second row: For the aggregation score, the pixel-level classifier score is summed over object-sized boxes. Each pixel represents the aggregation score of a bounding box whose upper lefthand corner is anchored at that pixel. Third row: Thresholded aggregation scores. Pixels marked white represent the upper lefthand corner of a bounding box deemed likely to surround an instance of the object. Fourth row: Aggregation scores after non-maximum suppression. Note that the image has been reduced to a small number of potential object locations. Fifth row: Pixels marked white are inside at least one of the bounding boxes which passes the aggregation score threshold (third row). These pixels are passed on for pixel classifier training at the next cascade phase.

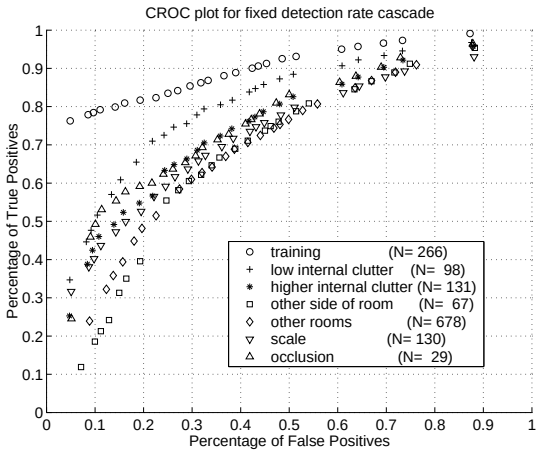
In summary, this experiment demonstrates that optimizing part classification and aggregation in a coupled way can effectively consolidate pixel classification results into gross image regions considered most likely to contain the object. This in turn makes object-level localization of the object easier.

5.5 Training Data Class Distribution

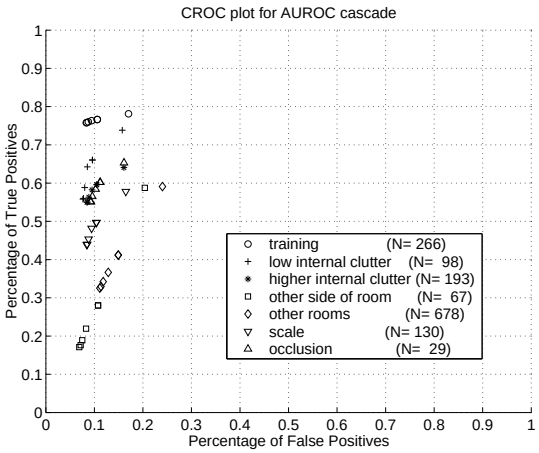
In the experiments in Section 5.3, the number of positive and negative examples used for training classifiers was equal; in particular, we sampled a fixed number of object pixels and background pixels from each image and used these examples to train decision trees and prune them. Here, we experimented with skewing the distribution of positive and negative examples at training time so that it matches the distribution of object and background pixels found in the training images. The rationale for doing so is our observation that, in the stool images, far more of the image is consumed by the background than the object, and that the variability in the appearance of the background is far greater than that of the object; therefore, we feel that at training time we should attempt to account for that high variability and high concentration of background through a larger number of training examples.

In Section 5.3, we extracted a fixed number k of object pixels, and k background pixels, from each image. Here, instead, we select m percent of the total number of object pixels from each image, and m percent of the background pixels. Since there are roughly 10 times more background pixels than foreground pixels in each stool image, this means that at training time there will be roughly 10 times as many negative examples than positive examples.

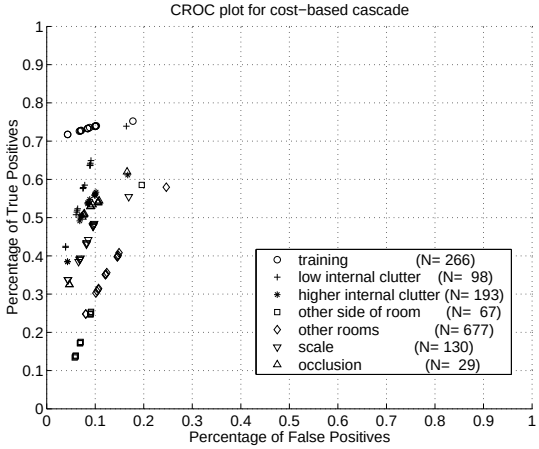
Decision trees induced using the skewed set of training examples performed poorly. That is, these decision trees were either eliminated entirely during the pruning procedure or failed to generalize well on new images. Using a balanced distribution of training examples during tree growing and a skewed distribution of examples during pruning worked quite well, however. Specifically, we selected k object pixels and k background pixels from each image in the tree-growing set and induced a decision tree as in Section 5.3. Then, we selected m% of the object pixels and m% of the background pixels from each image in the holdout set and pruned the tree. We trained a pixel classification cascade using parameters identical to that of Section 5.3, except for this modification. In the earlier experiment, 8000 object pixels and 8000 background pixels were sampled from each holdout image, corresponding to roughly 10% and 100% of the two categories of pixels. Here, we selected



(a) Fixed Detection Rate



(b) AUROC



(c) Cost-based

Image Type	Fixed		Cost		AUROC	
	TP	FP	TP	FP	TP	FP
Training	0.763	0.049	0.718	0.043	0.758	0.083
Low Int. Clutter	0.347	0.048	0.422	0.039	0.558	0.076
Higher Int. Clutter	0.253	0.047	0.384	0.043	0.550	0.085
Other Side Of Room	0.119	0.071	0.094	0.045	0.171	0.069
Other Rooms	0.239	0.089	0.248	0.080	0.325	0.111
Scale	0.316	0.051	0.337	0.044	0.439	0.084
Occlusion	0.245	0.051	0.325	0.047	0.551	0.092

(d) Summary

Figure 5.15: Figure 5.15(c) shows a Pixel CROC plot for a pixel-level classifier cascade identical to the one depicted in Figure 5.8, but using more negative examples relative to positive examples during pruning. Also shown are analogous CROC plots for cascades trained using AUROC-based (Figure 5.15(b)) and fixed detection rate (Figure 5.15(a)) pruning. Figure 5.15(d) lists the true positive and false positive pixel rates for the three cascades after the final cascade phase. Note that the cascades converge to more or less the same true positive and false positive rates as before, although the ones given more training data does so faster.

10% of all object pixels, and 10% of all background pixels, or roughly 8000 background and 800 object pixels, from each holdout image. Pixel classification results are shown in Figure 5.15. The cascade converges to very similar operating points as the cascade of Section 5.3, for example 60% true positive rate and 10% false positives on the low-internal-clutter images, but it does so much faster than the previous cascade. In particular, for this cascade most false positives are removed from the images within the first few phases, while for the previous cascade it took several cascade phases to reach similar operating points. The prevalence of negative examples in the holdout set helps to explain why the cascade screens false positives from the images so well; in particular, consider the sum of misclassification costs, $c_{fp} * fp_p + c_{fn} * fn_p$, being minimized at pruning time. In this experiment, since the number of background pixels is so much higher than the number of object pixels, there will be a greater prevalence of false positives than false negatives, in other words the ratio of fp_p to fn_p will tend to be higher than it was in the cascade of Section 5.3. This means that $c_{fp} * fp_p$ will tend to dwarf the contribution of $c_{fn} * fn_p$ to the sum of costs, and the classifier will try to decrease costs by decreasing false positives.

5.6 ROC Optimization Strategies

In this section we present our observations on the experimental performance of cascades based on fixed detection rate, cost-based, and AUROC-based criteria. We draw our conclusions from the aggregation-sensitive cascades discussed in Section 5.4, the cascades trained on skewed class distributions in Section 5.5 as well as edge probe cascades trained on the ladder images. In short, we learned that none of the three ROC optimization schemes dominated the others in terms of overall classification performance, and that cost-based and AUROC-based cascades performed very similarly in every experiment. In some cases, fixed detection rate cascades took many more iterations than AUROC and cost-based cascades to converge to more or less the same performance numbers, pointing to the flexibility which which the latter two methods are allowed to select operating points. However, as we report in Section 5.7, the performance of cost-based and AUROC-based cascades can be sensitive to how their misclassification costs are set.

We trained cascades of edge probes to discriminate between ladder edge pixels and background edge pixels in the lab images (Figure 3.13). In other words, unlike the previous experiments in this chapter, we classify binary edges with edge density features rather than classifying the entire image with edge operators. Neither were the feature selection

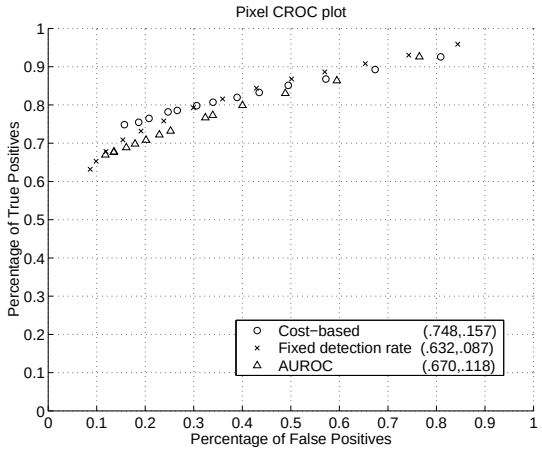


Figure 5.16: A CROC plot of pixel classification results on the ladder images from cascades trained using fixed detection rate, cost-based, and AUROC-based criteria.

techniques in Section 4.1.1 employed to reduce the number of image features at training time. We used 55 training images and 55 holdout images of the ladder to train cascade classifiers with fixed detection rate, cost-based, and AUROC-based optimization criteria. As in Chapter 3, we set the target detection rate for the fixed detection rate cascade to 98%, and misclassification costs for the other two cascades was set to $(c_{fp}, c_{fn}) = (.01, .99)$.

Edge pixel classification results for these cascades is plotted in Figure 5.16. The final results for all three cascades are similar– between 60% and 75% of object pixels are retained, and between about 10% and 15% of edge pixels among the background are false positives. The fixed detection rate cascade (x) achieves the lowest false positive rate at 9%, but its true positive rate, at 63%, is lower than it is in the other two cascades. A typical example showing pixel classification results at many cascade phases by the AUROC cascade is shown in Figure 5.17. Figure 5.18 shows the pixels remaining in contention after the final cascade phase for each of the three cascades applied to the image in Figure 5.17. For this image, the fixed detection rate cascade (bottom center image) is able to remove all the background pixels. However, for all three cascade types the results are similar: a high percentage of edge pixels among the background are eliminated, and a high percentage of edge pixels on the object remain.

Each of the three cascades behave similarly on the ladder at training and run time. In contrast, consider the pixel classification cascades depicted in Figure 5.15. The fixed detection rate cascade (Figure 5.15(a) makes piecemeal progress toward the goal of discarding all false positive pixels; it starts with the full set of all false positive pixels and each phase steadily reduces the size of that set by 5% or so. On the other hand, the cost-based and

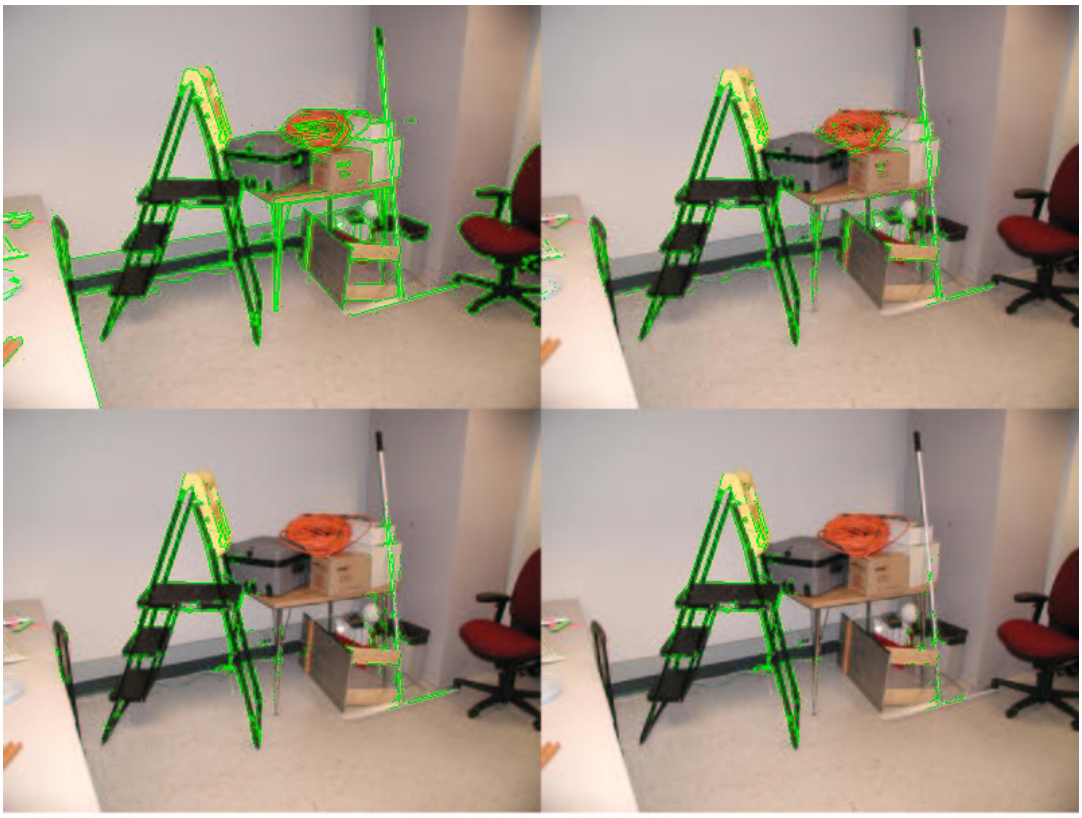


Figure 5.17: An example of edge pixel classification on the ladder at many phases. The input image with overlaid edges is shown at the upper left; pixel classification results after cascade phases 5,10, and 15 are shown left-to-right, top-to-bottom. Edges which have not been discarded by the cascade are shown in green.

(a) Input Edges

(b) Classification Results

Figure 5.18: A comparison of pixel classification results on a typical image from classifier cascades trained with cost-based (left), fixed detection rate (center), and AUROC (right) criteria. Input edges are shown above.

AUROC-based cascades (Figures 5.15(b) and 5.15(c) remove most of the false positives in the first phase, and eventually converge to roughly the same operating points as the fixed detection rate cascade, but in fewer phases. We see similar behavior in the aggregation filter results shown in 5.12: the phases in the fixed detection rate cascade make steady, even progress in removing false positives, while the cost-based and AUROC-based cascades stop at similar operating points, but faster. Our intuition for this behavior is that the cost-based and AUROC-based cascades have more flexibility in selecting operating points, as argued in Section 4.3.

5.7 Sensitivity to Parameters

In Algorithm 2 we present a cascade training procedure through which each classifier corresponds to an aperture of a particular size. As more classifiers are added to the cascade, the size of the aperture grows, so that later classifiers in the cascade have access to more visual features in order to discriminate object pixels from background pixels. However, the decision to allow only one classifier per aperture size is somewhat arbitrary; in particular, it is tempting to add a several classifiers per aperture size, only growing the aperture when the classifiers "run out" of discriminating features inside that aperture to take advantage

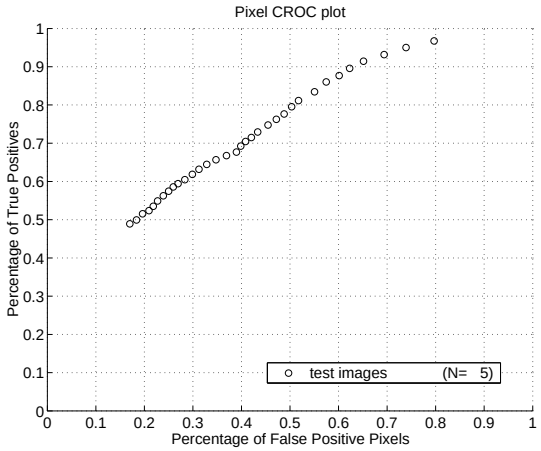


Figure 5.19: Pixel CROC plot for a pixel-level classifier cascade that overfits. For each aperture size, phases are added to the cascade until the classifiers fail to reduce the false positive rate. The resulting cascade does not generalize to new images as well as cascades which employ one cascade phase per aperture size. Note the final true positive rate near 50%, compared to 70% in the results listed in Figure 3.2.

of. Indeed, in the experiments presented in Sections 5.4 and 5.3, we add several classifiers for each aperture size, only increasing the aperture size when a particular classifier fails to increase its ROC scoring criterion by a given amount. For example, when using cost-based ROC optimization, if a particular classifier fails to decrease the sum of costs $c_{fp} * fp + c_{fn} * fn$ by more than a given threshold γ.

Unfortunately, the generalization performance of our pixel-level classifiers are sensitive to the setting of γ. In particular, if γ is so low that classifiers are added to the cascade even if they minimally improve their performance criterion, the pixel classification performance of the cascade on novel images suffers. An example of this phenomenon is illustrated in the pixel CROC plot shown in Figure 5.19. We trained a pixel classification cascade using the baseline algorithm presented in Chapter 3, but with one modification: rather than adding one classifier per aperture to the cascade, we continued to add new classifiers for each aperture size as long as the new classifiers reduced the false positive rate by any amount. The cascade was trained until it had added 34 classifiers over the first 8 aperture sizes, and was then applied to test images for evaluation. Figure 5.19 shows that the cascade achieves a true positive rate of about 50% with a false positive rate of about 18%; this is a significantly lower true positive rate at a similar false positive rate than the 74% recorded by a cascade which is identical except for the fact that it only adds one classifier per aperture size for a total of 20 cascade phases (see Figure 3.2, row 7).

The intuition for this performance dropoff is that the cascade with many classifiers per aperture is overfitting. That is, rather than discriminating between characteristics of the

image features which arise due to the underlying object parts, the classifiers are discriminating between chance characteristics of the image features that arise due to noise or small training set size. It is not surprising that adding an arbitrary number of classifiers to the cascade can lead to overfitting; similar ensemble methods like AdaBoost are now generally considered prone to overfitting if a large enough number of complex classifiers are added to the ensemble without taking measures to prevent overfitting. In the experiments involving aggregation-sensitive training and edge operator probes, we have been able to avoid overfitting by setting the threshold γ which discards classifiers if they fail to improve performance on the training set by a fixed amount. Unfortunately, setting γ is not necessarily intuitive at the beginning of an experiment, and doing so requires some trial and error. An alternative way to guard against overfitting is to partition the set of training images into two subsets: one set is used to train cascade phases, and the other is used to evaluate the performance of those phases. Cascade phases which perform poorly on the second set of images are not added to the cascade. This cross-validation approach to preventing overfitting has been applied to AdaBoost ensembles, for example in [105].

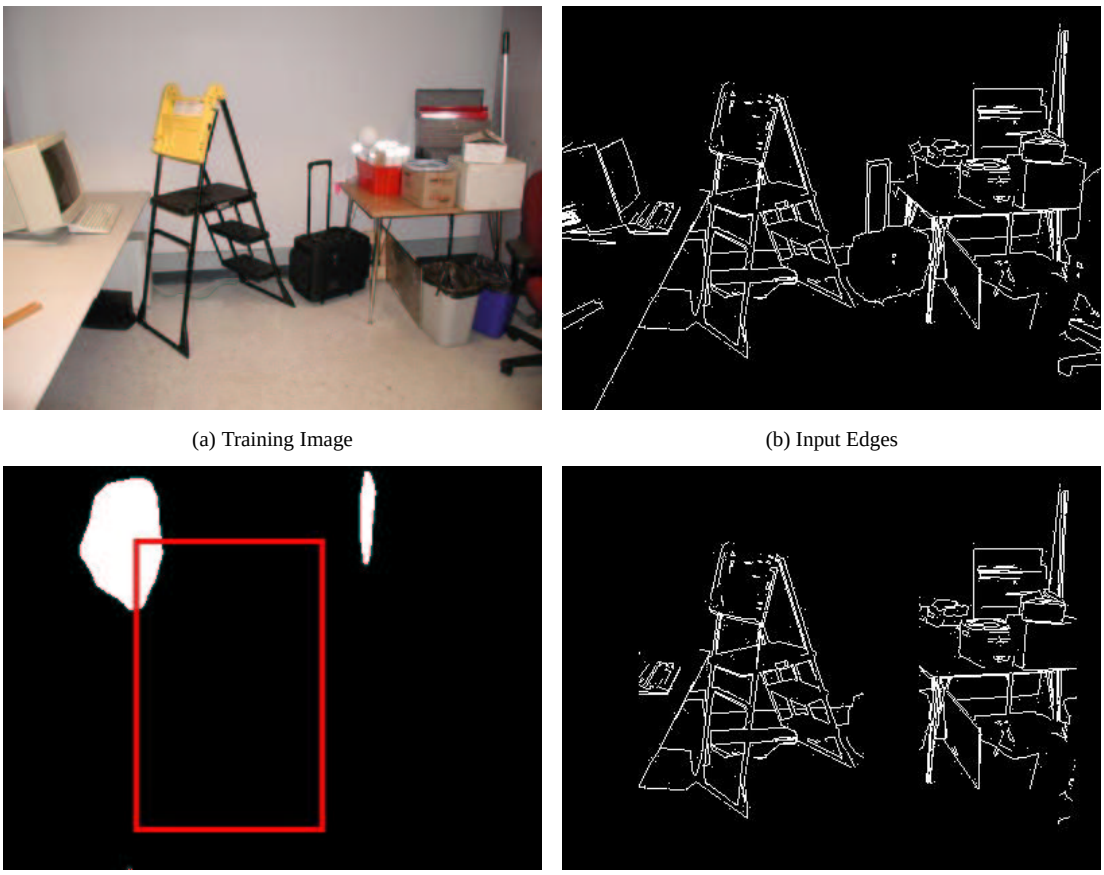
Another set of parameters which significantly affect performance in cost-based optimization of pixel-level classifiers are the costs for false positives and false negatives, c_{fp} and c_{fn}. As explained in Section 4.3, given a particular aggregation scheme, we can use labeled training data to estimate good values for c_{fp} and c_{fn} by generating synthetic part classification results corresponding to ranges of possible false positive and false negative rates, then running the aggregation phase on the synthetic data to evaluate how false positives and false negatives affect the accuracy of aggregation. However, in lieu of such experiments on the aggregation step, we must set values for c_{fp} and c_{fn} by hand. Improper settings of these costs can dramatically alter the training performance of the cascade. For example, imagine that the number of background pixels, np_-, is orders of magnitude larger than the number of object pixels, np_+, but that the cost of a false negative is not orders of magnitude larger than the cost of a false positive. Then a low-cost classifier will simply be one which classifies any and all pixels as background. Since there are relatively few false negatives fn_p in this case, and the cost of a false negative c_{fn} is not very high, the contribution of $fn_p * c_{fn}$ to the total cost $fn_p * c_{fn} + fp_p * c_{fp}$ will be insignificant, and since all pixels are classified as background, the number of false positives fp_p is zero. On the other hand, if c_{fn} is set too high, then the contribution of $fn_p * c_{fn}$ will dwarf that of $fp_p * c_{fp}$ to the total cost, leading to classifiers which classify any and all pixels as belonging to the object to avoid any false negatives. In practice, c_{fp} and c_{fn} must be set so that neither extreme situation arises, and this takes some experimentation. We emphasize,

however, that manually setting c_{fp} and c_{fn} is only necessary if the costs are not provided based on external factors like aggregation performance.

5.8 Limits to Aggregation-Sensitive Training

Figure 5.14 shows example results of classifying an image with a classifier cascade which employs aggregation-sensitive filtering. By aggregating raw pixel classifier scores and discarding whole sections of the image which do not give high object-level scores of containing the object, we are able to remove large sections of the image even after the first classification phase (see Figure 5.14(j)). Discarding many pixels as early in the cascade as possible is important for tractability reasons. Specifically, as we suggest in Section 3.6.5, the number of image features does increase as the aperture size grows, but computation time does not grow intractable because through each classifier phase we discard pixels, which effectively reduces the number of training examples. If early classifier phases fail to remove a significant number of pixels from consideration, it is possible that training later cascade phases could be overwhelmed by the large number of features and training examples.

Our approach to aggregation-sensitive part classifier training can lead to situations in which few pixels are discarded early on, especially if the size of the object box is relatively large with respect to the size of the image. Specifically, we retain pixels for future phases of pixel classification if they are contained in any of the bounding boxes that pass the aggregation score threshold; if the box covers a large portion of the image, then most of the pixels will be retained even if only a few possible bounding boxes remain in contention. As a result, even if the cascade makes progress at removing image locations from consideration at a box level, it might make little progress in reducing the number of pixels that the pixel classifier must process. An example of this phenomenon is shown in Figure 5.20. A classifier cascade was trained using edge probes on binary images as image features, and aggregation-sensitive part classifier training. After several phases, the cascade was largely successful at removing potential object bounding box locations from consideration in the training images. Figure 5.20(a) shows a typical training image and Figure 5.20(b) shows the edges detected in it. After several cascade phases, the image was reduced to the bounding box locations shown in Figure 5.20(c); specifically, the pixels marked white represent the upper left hand corner of a potential bounding box still under consideration by the cascade. Bounding boxes covering the ladder are still represented– they are in the white blob



(a) Training Image

(b) Input Edges

(c) Bounding Box Locations

(d) Remaining Edges

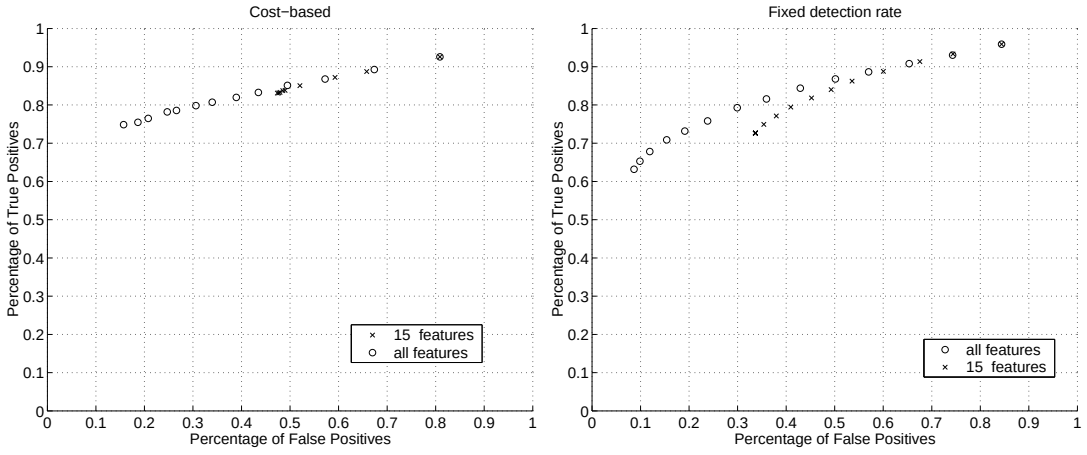
Figure 5.20: An illustration of the fact that aggregation-sensitive part classifier training tends to quickly discard large portions of the image in early cascade phases only if the object size is small compared to the size of the image. Figure 5.20(a) shows a training image, and Figure 5.20(b) shows the binary edges extracted from it. Figure 5.20(c) depicts the possible locations of object bounding boxes after several phases of classification with an aggregation-sensitve classifier cascade. Each white pixel depicts the upper lefthand corner of a viable bounding box for the object after many cascade phases. An actual bounding box for the object is shown in red to give a sense of the bounding box size. Note that most of the possible spurious bounding box locations among the background have been removed from contention, with the exception of the white blob to the right and some of the pixels in the lefthand blob. Figure 5.20(d) shows the edge pixels which land in at least one of the retained bounding boxes. Note that even though most of the spurious bounding boxes have been discarded, most of the edge pixels are still present.

to the left– as well as a sliver of false positive boxes in the blob to the right. At an object level, most false positives have been removed from the image; however, the number of false positive pixels still under consideration, shown in Figure 5.20(d), is still high and includes almost all of the background edge pixels in the image. Each of the pixels shown in Figure 5.20(d) are contained in at least one of the bounding boxes represented in Figure 5.20(c). In this case early phases in the aggregation-sensitive cascade fail to remove many background pixels because the bounding box is so large that it only takes a few false positive boxes to cover all the pixels in the image. Meanwhile, in the stool image, the bounding box size is so small relative to the size of the image that it is easier to remove whole sections of the image from consideration.

5.9 Feature Selection For Edge Probes

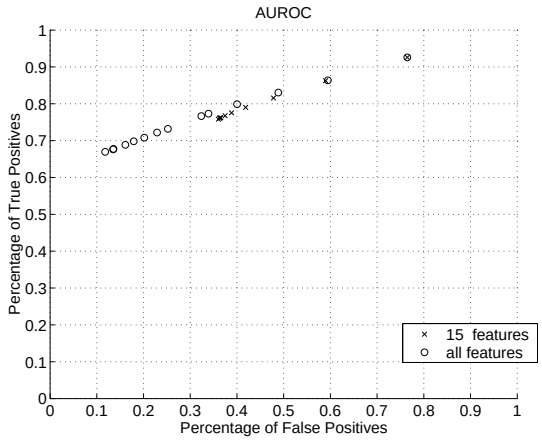
In Section 4.1.1 we present a filter method for selecting which operator-offset pairs should be used to train classifiers. Since the number of possible combinations edge operators and relative probe centers is large, feature selection in the context of edge operator probes is a matter of necessity. However, it is natural to ask whether feature selection can be useful for reducing the number of features used to train cascades of edge probes. Since the number of probe centers grows as the aperture size increases, it would be advantageous computationally to be able to train classifiers using features drawn from a limited number of probe centers. However, this experiment suggests that for the edge probes, which are inherently less expressive features than edge operator probes, good pixel classification performance requires that a large number of spatial features be available to the classifier at training time.

To show this, we trained two sets of classifier cascades, the only difference between the two being the number of features used to train the classifiers. In the former case, we used the full set of relative probe centers at each aperture size to train the classifier for that aperture; in the latter case, we used the filter method from Section 4.1.1 to pick the 15 most discriminative relative probe centers and train the classifier based on those features. We will call the first type of cascade the “all-feature” cascade and the second type of cascade the “15-feature” cascade. In both cases, we restricted the cascade to have one classifier per aperture size, due to the overfitting concerns raised in Section 5.7. We trained a total of 6 different cascades: for each of the three ROC optimization strategies presented in Section 4.3, we trained an all-feature cascade and a 15-feature cascade. Figure 5.21 contains one CROC plot for the fixed detection rate, cost-based, and AUROC-based cascades, showing



(a) Cost-based

(b) Fixed detection rate



(c) AUROC

Figure 5.21: Comparison of pixel classification performance for cascades given access to all features (shown "o") and those given a subset of 15 features (shown "x"). The features are edge probes, and the ladder image set is the target. CROC plots are given for cost-based (Figure 5.21(a)), AUROC-based (Figure 5.21(c)), and fixed detection rate (Figure 5.21(b)) cascades.

the pixel classification rates for all-feature cascades ("o") and 15-feature cascades ("x"). For the first few cascade phases, the true positive and false positive rates are nearly identical for the two types of cascades because the total number of probe centers is less than or equal to 15. But as features from more relative probe centers become available to the all-features classifier, it continues to discard false positives, while the 15-feature classifier ceases to make progress.

Thus, pixel classification cascades based on edge probes do not perform well when feature selection reduces the number of features available for training, while our results in Sections 5.4 and 5.3 show that classification cascades based on edge operator probes perform well when only 5 features are available to each classifier. Our intuition for this is that

edge operator probes are more expressive than edge probes; besides signaling the presence or absence of edges in a particular region, edge operator probes signal edge orientation and scale characteristics. In so doing, they provide more discriminative power to the classifier in a smaller number of features.

5.10 Conclusions

These experiments have illustrated a number of characteristics of our basic shape-based localization algorithm from Chapter 3 as well as the extensions presented in Chapter 4. First and foremost, the experiments show that we can automatically optimize image features, part classifiers, and aggregators which do an effective job at correctly identifying object pixels and overall instances in test images taken in similar environments to the test images. They also show that training the part classifiers and aggregation filters in tandem can lead to more effective, more predictable localization performance than if the two recognition steps are optimized separately. We also see from these experiments that the classifiers we derive do not generalize perfectly to novel test environments that diverge greatly from those present in the training images. Finally, our cascades can be prone to poor performance if certain parameters of the cascade construction process are not properly set.

Chapter 6

Discussion

This chapter discusses previous and future work related to the techniques presented in Chapters 3 and 4.

6.1 Related Work

In this section we review work related to some key characteristics of our algorithms. In particular, we address the use of cascades and related methods for building classifiers out of components; we describe work in computer vision and neuroscience which relates closely to our paradigm of sparsely probing the image; and we show how aperture problems arise in other fields and the similarity between our aperture-growing scheme and related approaches.

6.2 Cascades and Ensembles

One of the major contributions of this thesis involves performing object recognition by applying a sequence of tuned image processing modules to the image. We refer to computational architectures of this type as *cascades*. Researchers in machine learning, computer vision, and neuroscience have each addressed the use of sequential conjunctions of computational modules for modeling and classification; however, researchers in each field have attached distinct connotations to the term "cascade." In this section we delineate the different senses of the term in order to clarify our use of it, and compare our cascade technique

to those of closely related object recognition algorithms.

Recently, several authors [118][68][58] have presented algorithms which use direct methods to estimate the image locations of objects based on texture cues, and describe them as cascades. In each of these approaches, a sequence of simple classifiers is trained to discriminate image patches corresponding to the target object from image patches projecting onto the background; a run-time image is processed by applying the simple classifiers to it in a sequence. More specifically, these approaches are distinguished by two characteristics. First is that training proceeds in a "sieve" fashion: training data points classified by the first simple classifier as an instance of the target object are used to train the second simple classifier; training data classified as clutter by the first simple classifier are discarded. In this way, the training procedure reduces the training set so that the k-th classifier in the sequence focuses on discriminating object data points from clutter data points which have not been identified as such by the first $k-1$ simple classifiers. The second defining characteristic of these approaches is *early stopping* at run-time: that is, evaluation of a particular test image point stops if the k-th simple classifier classifies the point as clutter. This run-time scheme, which effectively arranges the simple classifiers in a decision list [97], accounts for the impressive speed of these algorithms; in many cases, most of the test image is quickly removed from further consideration by the first few simple classifiers, leaving the bulk of the computation for those portions of the image most similar to the target object.

In contrast, classification architectures referred to as cascades by researchers in the machine learning community [41][28] possess neither of these characteristics. A series of simple classifiers is trained sequentially; however, each simple classifier in the series is trained using all instances of the training data rather than a reduced subset. In both schemes, the class probabilities output by earlier classifiers in the sequence are concatenated onto the end of each training instance as though they were additional input features, and the later classifiers in the cascade are trained on the augmented training set. At run time, each test instance is classified sequentially by each and every simple classifier in the sequence. While these cascade schemes do not feature a capability for early stopping, it should be noted that they were motivated by concerns for training efficiency, not run-time speed. In particular, cascade-correlation neural networks [28] were presented as a way to incrementally add hidden units to a neural network in a way that precluded the costly communication of error signals between many hidden units during training.

The term "cascade" carries a slightly different connotation in the neuroscience literature. There, the processing of visual signals is modeled in terms of distinct computational

modules which act in sequence of in parallel. However, models which arrange these modules in a linear series along the lines of the cascades outlined above are referred to as *discrete staged* accounts; cascade processes refer to the parallel, asynchronous processing of the visual signal by several different computational modules. For example, in [52], the process of picture naming is modeled in terms of three types of computational modules: vision modules which associate appearance properties of the image with appearance properties of known objects; semantic modules which contain information about the function of and associations between objects; and phonological modules which map the semantic meaning of the object onto a representation of its name. In a discrete staged model, the image is first processed by the vision modules, the resulting object identities are passed to the semantic modules, and the resulting semantic information is passed on to the phonological modules. In a cascade model, however, the image triggers the vision modules, and after that any number of phonological, semantic, and vision modules may be actively computing at any point in time.

Cascade methods for classification constitute a special subclass of the more general category of *ensemble methods*. Ensemble methods build up a complex classifier from a collection– an ensemble– of simpler classifiers. Like the boosting family of algorithms[101], our cascade technique falls into the category of methods for which the training of simple classifiers is necessarily sequential; that is, the training data for the kth simple classifier in the sequence has been modified in some way by the first $k-1$ simple classifiers that have been trained. This is in contrast to techniques such as stacked generalization[124] or bagging[17], in which a collection of simple classifiers may be trained in parallel and combined afterwords. Furthermore, our cascade may be grouped with other ensemble methods like cascade correlation and cascade generalization[41] which sequentially evaluate the simple classifiers in the order in which they were trained; in other ensemble methods, including boosting and bagging, the simple classifiers may be applied to the test example in an arbitrary order.

Prior classifier cascades for object recognition typically train each cascade phase to optimize a fixed detection rate criterion as we describe in Section 4.3. However, these previous techniques differ from ours in the design of the classifiers at each phases and the details of how they are assembled into cascades. In particular, in the Viola-Jones face detector [118] each cascade phase is an ensemble of simple classifiers. As in our approach, each cascade phase in Viola-Jones is trained using a combination of classification-accuracy-based and ROC-based criteria; specifically, at each phase, simple classifiers are added to the ensemble via AdaBoost [101] in order to maximize classification accuracy, but the threshold

function and training procedure for the cascade phase is affected by a fixed detection rate criterion. Specifically, simple classifiers are added to the cascade phase until the phase achieves an operating point (fp, tp) corresponding to user-set target false positive and true positive rates fp and tp. Li *et al* [68] and Lienhart *et al* [69] follow similar strategies, except that Li *et al* also incorporate a cost-based criterion to remove simple classifiers from cascade phases, and Lienhart *et al* produce a tree, rather than a linear sequence, of cascade phases. Meanwhile, Schneiderman [105] trains a single ensemble using AdaBoost but classifies image locations using a subset of simple classifiers at run time. Antifaces [58] can be thought of as building a cascade of linear discriminants, each of which are optimized according to fixed detection rate criterion. Finally, Amit *et al* [3] and Rowley [98] each train two-phase classifier cascades for face detection, and each phase sets thresholds according to fixed detection rate criteria. It should be noted that misclassification costs can easily be incorporated into boosting-based cascade phases by replacing AdaBoost with the very similar AdaCost ensemble trainer [30].

6.2.1 Sparse Image Probing

In this thesis we emphasize object recognition techniques which classify a test pixel by *bottom-up image probing*: accessing a sparse set of pixels in the vicinity of the test pixel and evaluating image features there. Neuroscience researchers have discovered that humans follow a similar strategy of accessing a series of sparse, isolated image locations while interpreting a view of a scene. This exploration of the image, referred to as a *saccade*, is accomplished via fast eye movements which focus the fovea, the high-resolution center of the retina, on portions of the scene which are the most informative for the visual task at hand. Image probing aims to achieve an analogous goal; we wish to focus feature computation on only those pixels which will be the most informative for the classification of the test pixel.

Several researchers have formulated computer models which account for, or attempt to mimic, human eye movements during visual cognition [94] [44][33][29]. What differentiates them from bottom-up image probing, however, is that they pursue a top-down interpretation of the scene. For example, in [94], the image is convolved with a bank of coarse-scale filters, resulting in a feature vector for each coarse location in the image. These feature vectors are correlated with stored feature vectors for images of target objects, and visual attention is moved to a local maximum, or local soft maximum, of the resulting correlation image. The image neighborhood surrounding the new attention point is con-

volved with a bank of finer-level filters, and attention is again moved to an image location which represents a maximum or soft maximum of similarity between image feature vectors and stored feature vectors for target objects. In this way, a coarse-to-fine exploration of image attention points is initiated, until at the finest level the attention lies on the target object itself. Similarly, in [44][33][29], features are extracted from the entire image, and attention is shifted to those parts of the image whose features form salient perceptual groups or are similar to those of target objects at a coarse scale. At those attention points, computation of features takes place at a finer scale, and attention is shifted again.

In contrast, through bottom-up image probing we scan through each pixel in the test image and evaluate fine-scale features in the vicinity of each test pixel; in effect, we shift "attention" to image locations in the neighborhoods surrounding test pixels in order to classify them. We view top-down saccade algorithms and our bottom-up image probing as complementary processes; it is possible to imagine a vision system which first follows a coarse-to-fine strategy to shift attention to interesting portions of the image, and then executes a series of image probes at the finest scale to determine which pixels in those image portions map onto the target object. Faisal *et al* [29] proposes a neuromimetic system which pairs a coarse-to-fine attentional mechanism with a fine-scale object recognition module; however, the recognition module extracts Gabor features from a predetermined, dense grid of image locations in the attentional area, rather than from isolated, sparse pixels. Giefling *et al* [44] propose an object recognition system which first detects salient areas in the image using texture segmentation and interest operators, then accumulates evidence for the presence of objects in the image by associating salient image areas and spatial relations between image areas with known objects.

Geman *et al* present a recognition approach that is very closely related to ours in terms of sparse image probing [2][3][42]. They induce decision trees which test local feature values at isolated image locations in order to classify training examples. Each node in the decision tree only searches for splits among a random subset of the total number of features, leading to fast tree induction. The key differences between our techniques and theirs are, first, that their features are usually identified with intensity or texture measures at small groups of image pixels, and second, that we compute features over growing apertures in a cascade format.

6.2.2 Aperture Problems

As mentioned in the previous section, we attempt to classify image pixels by evaluating local image features at isolated image locations in the neighborhoods of those pixels. Two obvious questions follow from this formulation. First, how should local image features evaluated in an image neighborhood be fused together into a final estimate of whether or not the test pixel projects onto the target object? And second, what should the size and shape of this neighborhood be? If the neighborhood is too small, the features evaluated in the neighborhood may be ambiguous; there may simply be too little visual information inside the neighborhood to reliably determine whether the pixel in question belongs to the target object or the clutter. On the other hand, if the neighborhood is so large that most of the pixels in it always project to the clutter, then most of the image features we evaluate will always be drawn from clutter, making it difficult to disambiguate clutter pixels from object pixels. Furthermore, classifying test pixels based on evaluating many image features over large neighborhoods may be computationally prohibitive.

The problem of how to integrate ambiguous local measurements over an extended image neighborhood into reliable estimates of local image properties is a special case of a widely known problem in computer vision known as the *aperture problem*. In general terms, the goal is to estimate a physical property at a set of image *sites*, *i.e.* pixels or patches of pixels, based on local measurements taken at each of those sites. Because the measurements are local, they are inherently ambiguous; thus, when forming an estimate for a particular site it becomes necessary to integrate local measurements taken from nearby sites. The aperture problem, in its most general form, is the problem of how to fuse those local measurements into site estimates. In our case, the sites are pixels; the ambiguous local measurements are edge features; and the property to estimate at each pixel is whether or not that pixel projects onto the target object.

Motion estimation in animals and computer systems gives rise to the version of the aperture problem most commonly addressed in the neuroscience and computer vision literatures. Given a video sequence of two or more images, the property to estimate at each pixel of the images is a motion vector that aligns the pixel to its corresponding pixel in the next frame. See, for example, [120] and [86] for discussions of motion aperture problems in biological vision systems and [4] for a discussion of how aperture problems can cause errors in artificial motion estimators. A related aperture problem arises in the problem of finding point correspondences between images in a stereo pair. Correspondences are found by measuring similarity between patches surrounding putative point matches in

the images, and the aperture problem consists of determining a patch size which resolves ambiguity without introducing too much computation or confusion.

Several solutions have been presented to these aperture problems. Neural network architectures were trained using supervised or unsupervised learning criteria to integrate motion estimates over a pre-determined neighborhood of nodes [107][108]. Veksler [115] proposed an algorithm which automatically determines the sizes and shapes of corresponding image patches in the two images, thus solving the aperture problem by integrating pixel-to-pixel similarities over image patches that maximally agree with each other. Earlier, similar approaches to the stereo problem grew and shrunk the patch sizes incrementally [57]. Freeman *et al* [39] produce local motion measurements at each image site, an integrate measurements from neighboring sites together using an iterative algorithm [39]. At each iteration, neighboring sites modify their motion estimates based on the motion estimates of neighboring sites, which are themselves being modified at each iteration. Thus, by the kth iteration, the motion estimate at a particular image site has integrated the estimates from all image sites k or fewer steps away.

The solution we take to our classification aperture problem is motivated by procedures such as Freeman's which iteratively integrate measurements over larger and larger neighborhoods in order to make reliable local estimates. In particular, we embed this intuition into a cascade-of-classifiers paradigm; each successive classifier in the cascade is trained to evaluate image features over successively larger neighborhoods in order to classify the pixel in question. If the image features in a particular image neighborhood are sufficient to determine with high certainty that a particular pixel projects to the background, computation stops and the pixel is classified as clutter. If, however, the image features do not identify the pixel as clutter, the neighborhood is grown and image features are evaluated over the larger neighborhood. In this way, the neighborhood is iteratively grown until the pixel is determined to be clutter or until it is determined to belong to the object by the final classifier in the series.

6.3 Future Work

In this section we present some directions for future research. The key issues which should be addressed are the simplified object representations employed by the part classification and aggregation steps, as well as the challenges presented by large numbers of features and images.

6.3.1 Spatial Representations of Object Parts

Our method for part classification based on edge cues consists of representing all pixels from the object as belonging to a particular "object" category, and all pixels from the background as belonging to a "background" category. The object category encompasses the appearances of all sections of the object at all poses, and therefore the image features drawn from this category can be be highly varied. Implicitly, when we try to discriminate object pixels from background pixels, we are forced to overcome this complexity. In the experiments we show, we are largely successful at discriminating object from background even though we place object pixels from entirely different parts of the object into the same category, but it is possible that with very complex-shaped objects, the distribution of image features may be so complex that it will be impossible for classifiers to extract meaningful regularities in the features which discriminate them from features drawn from clutter.

Ideally, instead of compressing all object parts into one omnipresent category, we would like to represent objects as collections of distinct parts, and model the distributions of image features corresponding to distinct parts discriminatively. This is the approach taken in Chapter 7; objects are represented as a set of categories corresponding to the appearances of distinct physical parts, and the goal is to determine which portions of the image feature space correspond to which parts. For recognition of the ladder, for example, instead of discriminating between object and background pixels, we could formulate the problem in terms of discriminating between image features corresponding to the legs, steps, other parts of the ladder, and background. Thinking of objects as collections of distinct parts allows more sophisticated aggregation schemes than the ones we present in Sections 3.5.5 and 4.2; for example, after the part classifier labels pixels as belonging to the legs and steps of the ladder, the aggregation step can constrain these labeled parts so that they appear in plausible arrangements with respect to each other (as in [34] for example). Unfortunately, this implies richly labeled training data; the legs, steps, and so on would need to be labeled in each training image, and in our examples our ladder images are only labeled with which pixels belong to any part of the ladder, and which do not. Labeling each training image in this way can be tedious, as we mention in Section 7.5.

Our shape-based recognition technique collapses all physical parts of the object into one category, giving rise to highly complex feature distributions; on the other hand, our texture-based technique represents all parts of the object as individual categories, giving rise to a labor-intensive data labeling task. A compromise between these two approaches is to represent objects in terms of multiple categories, each of which do *not* necessarily

(a) Object With Sub-boxes

(b) Probe centers for sub-boxes

Figure 6.1: A proposal for representing objects in terms of the appearances of parts appearing in sub-boxes. Figure 6.1(a): Instead of representing the object in terms of a single category of pixels, we represent each sub-bounding-box as its own category and train classifiers to discriminate between pixels lying in distinct sub-boxes. Figure 6.1(b): We can use spatial information to compute edge probes at probe centers that are more likely to lie on the object rather than the background. That is, for pixels lying in the upper right sub-box, probe centers below and to the left are more likely to lie on the object.

correspond to individual physical parts. Specifically, consider drawing a bounding box around the object of interest and partitioning the box into sub-boxes (Figure 6.1(a). We can allow each of the sub-boxes to correspond to a distinct category, so that at training time we attempt to discriminate between image features corresponding to pixels in distinct sub-boxes, and background. This converts our binary object-vs.-background classification task into a a multi-class discrimination problem, for which we must employ some sort of multi-class discriminant, for example [48] [123].

This approach has potential advantages. First, the sub-boxes do not correspond to physical object parts– for example, as the ladder rotates, object parts will appear in different sub-boxes– but characterizing pixels as belonging to particular sub-boxes has the potential to enable aggregation mechanisms which can incorporate constraints on the spatial arrangements of pixels classified as belonging to particular sub-boxes. A pixel classified as belonging to a sub-box in the lower lefthand portion of the object bounding box is expected to be surrounded by other pixels in the same or adjacent sub-boxes for example. Also, we can use this representation to fine-tune our image features so that we compute image features more likely to lie on the target object. An example is shown in Figure 6.1(b): since the point marked "+" is known to lie in the upper righthand sub-box, it makes sense to only compute edge probes at probe centers below and to the left, since these probe centers are more likely to lie on the image of the target object. Finally, while using sub-boxes boosts our descriptiveness of object parts, they do not imply any additional labeling tedium, since the sub-boxes are automatically computed from the object bounding box. Future work

should address the usefulness of this representation.

6.3.2 Spatial Aggregation

Our part classifiers are not the only step in our bottom-up localization process that pays no attention to how parts are spatially arranged in the image. The aggregation steps presented in Sections 3.5.5 and 4.2 sum up the number of points in a bounding box, or the sum of scores in a bounding box, without representing how the points or the scores are distributed spatially within the box. This leads to a variety of errors. For example, in Figure 5.14(c), most of the false positive pixels in the image have been discarded, and the only ones remaining cover a roughly object-sized block that is still "on." The aggregation scores (Figure 5.14(e)) indicate high scores for bounding boxes which only cover a percentage of the "on" area because it has no notion that large contiguous sections of zero-scoring pixels indicate that the object is not likely to be present. A straightforward improvement to our aggregation step, which should be explored in future work, consists of considering the aggregation filter as a mask M which we correlate with the pixel classifier scores:

$$ag(b_{\boldsymbol{\theta}}, \mathbf{q}, W) = \sum_{x \in [\frac{-b_{\boldsymbol{\theta}w}}{2}, \frac{b_{\boldsymbol{\theta}w}}{2}]} \sum_{y \in [\frac{-b_{\boldsymbol{\theta}h}}{2}, \frac{b_{\boldsymbol{\theta}h}}{2}]} W[\mathbf{q}[x] + x, \mathbf{q}[y] + y] * M[x, y]$$

The values in the mask can be estimated from pixel classification scores on training data. In future work, aggregation steps like this which take account for spatial distributions of parts should be investigated.

6.3.3 Large Data Set Problems

One of the main drawbacks of our classifier cascade is that it is extremely computation-intensive. As we report in Chapter 3, it can take a day or more to produce a 20-phase cascade, and in none of our examples is the number of training images very large: from roughly 100 for the ladder experiments to 200-350 for the stool experiments. To broaden the applicability of our technique to large collections of training images covering wide variations in variables like pose, future work should produce computational speedups which address the bottlenecks caused by large volumes of training data.

For example, when training classifiers, we randomly sample a large set of pixels from each image even though many of the pixels will be redundant with respect to image features.

When employing edge operator probes (Section 4.1), we train classifiers based on pixels drawn from all over the image, even though most of the pixels in edge-less portions of the image will have near identical (*i.e.* zero) responses to all of the edge operators. Quickly removing redundant training examples will be key in applying our technique to larger data sets.

Decision tree induction also has the potential for significant speedups. In particular, at each node in the decision tree we search for good split values in each attribute brute force, by considering each training example in turn and computing a "goodness-of-split" criterion for a split just before or just after the value for that example. Obviously the complexity of brute-force search for split values grows linearly with the number of training examples, so in the future we should employ smarter techniques for searching for good splits, for example by hierarchically clustering the attribute values and evaluating splits between the clusters. Also, we can speed the process of building trees from a large number of features by incorporating randomization techniques as in [2]. That is, rather than exhaustively searching through the set of all features for a good split at each node, we select a random subset of features and search for good splits amongst them.

6.3.4 Automatic Selection of Probe Centers

One weakness in our approach is that the layout of relative probe centers is chosen by hand. In particular, our policy of spacing the relative probe centers in rings spaced σ apart is somewhat arbitrary. While this particular spacing of spatial features works well in our experiments, it is unclear *a priori* that this set of image features contains sufficient information for discriminating object pixels from background pixels. On the other hand, it is possible that our layout of relative spatial features is overly dense; in other words, we may be able to perform our discrimination task with features more sparsely spread out over the aperture. Since the number of image features grows with the aperture size (See Table 3.5 for example), it is especially desirable for the relative probe centers to be as sparse as possible for computational efficiency, while containing a sufficient amount of features to discriminate reliably. Therefore, the impact of the layout of relative probe centers on localization performance should be addressed in future work.

In particular, one avenue for future research is the automatic selection of probe centers. Recall that in Section 4.1.1, we train a decision stump for each operator-offset pair $(g_{\boldsymbol{\theta}}, \boldsymbol{\delta})$ and assign a score $n(g_{\boldsymbol{\theta}}, \boldsymbol{\delta})$ based on how well the decision stump discriminates object pixels from background pixels. Operator-offset pairs with high scores are considered favorable

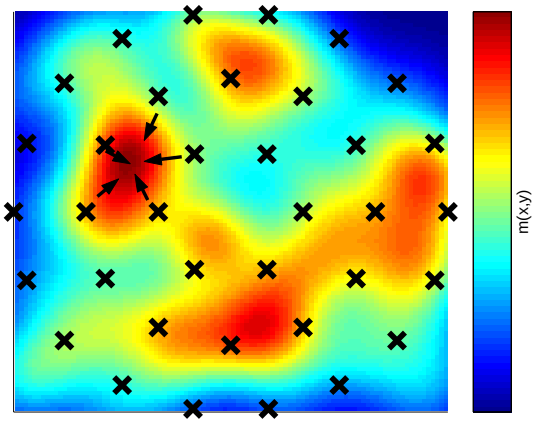


Figure 6.2: Proposal for automatic selection of relative probe centers. We start with an initial grid of relative probe centers (marked "x"), and a function $m(x)y$ which measures how well edge probes evaluated at spatial offset (x, y) discriminate between object and background pixels. Points with low To find a discrimination-optimal set of relative probe centers, we could move the initial set of centers so that they maximize m, for example by gradient ascent. Probe centers in the vicinity of the same mode (for example, the ones with arrows attached) would converge to the same spatial offset.

for training full decision trees because they do a good job of discriminating on their own. Consider a function $m(\boldsymbol{\delta}) = \sum_{\boldsymbol{\theta}} n(g_{\boldsymbol{\theta}}, \boldsymbol{\delta})$ which summarizes the discriminability scores for all the operator-offset pairs at a relative probe center located at offset $\boldsymbol{\delta}$ with respect to the query point. Prior to training any classifiers, we can use m to drive the design of the set of relative probe centers we train the classifiers with; namely, we can use optimization procedures to find a set of relative probe centers $\{\boldsymbol{\delta}\}$ such that $m(\boldsymbol{\delta})$ is high for all of them.

Figure 6.2 illustrates how m can be used to induce an efficient, informative set of relative probe centers. Starting with an initial set of relative probe centers $\{\boldsymbol{\delta}\}$ (marked "X"), we use gradient ascent to move each $\boldsymbol{\delta}$ toward the modes of m; in other words we iteratively move each $\boldsymbol{\delta}$ in the direction $[\frac{\partial m}{\partial x}, \frac{\partial m}{\partial y}]$. Relative probe centers near the same mode (for instance, the ones with arrows attached in Figure 6.2) converge to the same spatial location and all but one of them can be removed from the set of probe centers.

This approach has the potential to be computationally difficult however. The decision stump criterion m is discontinuous with respect to the location of $\boldsymbol{\delta}$. In particular, a small change in $\boldsymbol{\delta}$ may cause an entirely different "best split" threshold and therefore an entirely different discrimination criterion for that split. Thus, in order to do gradient ascent, an approximation of or replacement for m is needed such that the derivatives $\frac{\partial m}{\partial x}$ and $\frac{\partial m}{\partial y}$ are everywhere well defined, and efficient to compute. Otherwise, it is possible to apply stochastic optimization procedures to find a discrimination-optimal set of image features, as in [70]. Future work should address these issues.

Figure 6.3: An illustration of the fact that edge probes are recomputed on the remaining edge pixels after each cascade phase. The input edge image is shown in the upper left; edge probes evaluated over the whole edge image before cascade phases 2,5,8,11, and 15 are shown left-to-right, top-to-bottom.

6.3.5 Feature Recomputation

One unique characteristic of the training and run-time procedures in Algorithms 2 and 3 is that image features are recomputed after each cascade phase, based on the edge pixels remaining under consideration. That is, during each cascade phase, for each pixel $\mathbf{q}$ which has not been discarded, we compute edge probes $ep(\mathbf{q} + \boldsymbol{\delta}, \mathcal{G})$ which measure local edge densities among the set of all pixels $\mathcal{G}$ which have not been discarded. As pixels are discarded from the image, local edge densities drop, meaning that $ep(\mathbf{q} + \boldsymbol{\delta}, \mathcal{G})$ may take on different values at each cascade phase. An example of this fact is shown in Figure 6.3: edge probes at all locations in the image are computed at different cascade phases.

Thus, unlike other recent cascade approaches to recognition [118][68][58], our set of image features is non-stationary; the same image features take on different values at different cascade phases. An alternative is to compute edge probes on the full set of edges and use those feature values during each cascade phase; doing so would save the computation required to recompute features at each phase. Initial, anecdotal experiments suggest that re-computing edge probes causes a significant boost in discrimination performance, but future work should address when, and why, this is the case.

Chapter 7

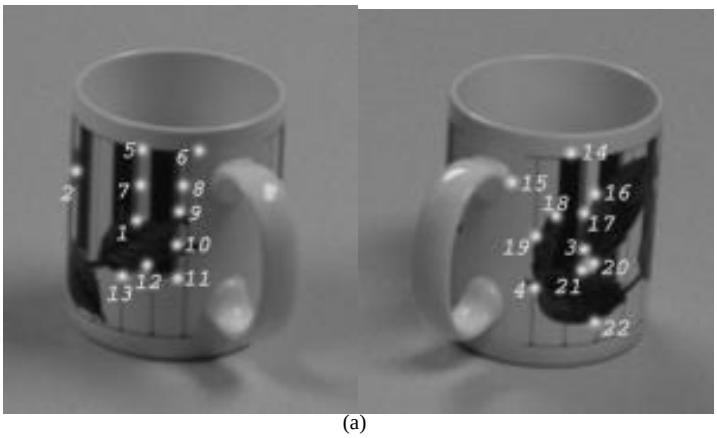
Discriminative Image Filtering

In this chapter we focus on image features for bottom-up localization based on texture cues. Recall that in our formulation, the first step of bottom-up localization consists of projecting image patches onto a low-dimensional feature space using a feature mapping function $f()$. Here, we present an approach for arriving at $f()$ in such a way that the image features corresponding to distinct object parts and clutter have disparate values. In particular, we convolve the image patches with a set of filters and linearly combine the filter outputs in such a way that the outputs are discriminated for distinct parts.

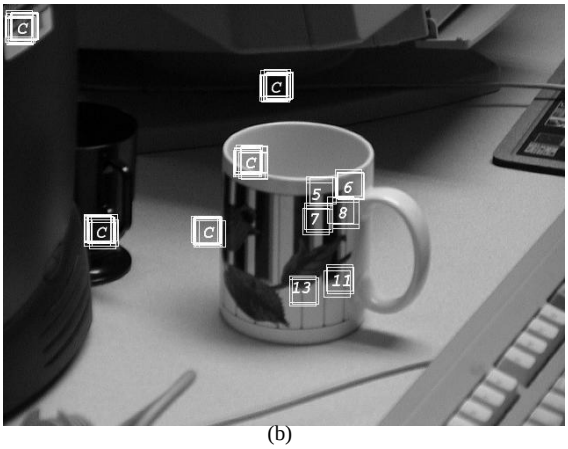
7.1 Introduction

In many texture-based approaches to recognition, image patches are represented by their responses to one or more filters. A common paradigm is to first gather up example image patches containing object parts and run them through a bank of filters; the filters can then be applied to a novel image and their responses compared to those for the training views to accomplish part classification. Filter sets appropriated for this purpose include convolutional kernels like Gabor wavelets [10] [81][102] and Gaussian derivatives[102][10][93], differential invariants based on combining the outputs of those kernels [103], local eigenspaces/PCA [22][56][84], and color statistics[110]. Each contains distinct characteristics that make them advantageous; for example, Gabor kernels are well-localized in space and frequency, invariants can tolerate transformations of the image, and local eigenspaces minimize reconstruction error of the training views.

But none of these filters are designed from the beginning with part classification in



(a)



(b)

Figure 7.1: Figure 7.1(a): Example images with object parts labelled. Figure 7.1(b): Detection of a subset of the parts from Figure 7.1(a) and clutter (marked "C") in a novel image. For display purposes, we only searched for six of the 11 labelled parts on this side of the mug. The number of rectangles around each patch is proportional to the confidence in its classification. Note that the window on the upper left portion of the mug is *not* mislabelled as clutter; since none of the parts of interest are on that portion of the object, it is technically "clutter" for parts classification purposes. Results for classification of the complete set of mug parts using the approach described in this paper are presented in Section 7.4.

mind. Those relying on pre-defined banks of kernels or statistics are not necessarily tuned to the appearances of specific objects and environments, so if the design parameters of the kernels are not properly set, it is possible that filter responses from distinct object parts will be indistinguishable from each other, or that outputs for background clutter will be the same as those for the object to be searched for. Local eigenspaces, although derived directly from training views, are not necessarily tuned for discriminability either– they maximize the pooled covariance of all example patches of all object parts, but do not necessarily encourage discriminability between the different parts and background. As a result, implementations rely on two types of optimizations: tuning the design parameters of the filters so that outputs for distinct parts are distinct, and adjusting the settings of the part classifier that decides which filter outputs at run-time correspond to which parts of object. For example, suppose we want to recognize objects by using image responses to a set of Gaussian derivatives to tally votes for object parts in a hash table, as in [93]. For some recognition scenarios it may be unclear how to determine what Gaussian standard deviation (later referred to as the "width" or "scale" of the kernel) will result in responses that are well-clustered for the same part and well-separated for different parts; it may also be difficult to determine how to size the bins in the hash table to minimize incorrect votes.

There are two common solutions to this problem. First is to discretize the range of reasonable filter parameter settings, run recognition experiments using each setting in turn, and select the setting which gives the best performance. Second is to generate filter responses over many parameter values for the same image patch at training time and/or run time. As an example of the second approach, Schiele *et al* [102] gather responses of training patches to Gaussian derivatives or Gabor kernels at several scales off-line and compare these to the outputs for a single scale on a new image. Local eigenspace techniques, on the other hand, tend to take the first approach, generating filter responses for a single parameter setting for training and testing[22][56][84].

We show experimentally that by optimizing linear combinations of filter sets over a range of filter parameters we can achieve good part classification without requiring a suite of trial-and-error experiments or training a part classifier with multiple distinct sets of responses per patch. Furthermore, we demonstrate that in some cases, the resulting filters can enhance classification to a degree that enables simpler classification mechanisms at run-time. We emphasize that we do not propose a new functional form for image filters; rather, we introduce a way to use training data to automatically combine sets of filters of any type so that the image feature mapping enhances part classification.

We call the resulting image features *discriminant filters* because the combining coefficients are optimized to discriminate between responses for distinct object parts and clutter. As an example, to design the Gaussian derivative kernels mentioned earlier we would synthesize many filter sets, one for each choice of Gaussian width over a range of plausible values. We then compute the responses of the training patches to each of the filter sets, and determine what the coefficients of a linear combination of those filter sets should be so that the responses for different object parts are discriminated from each other, while responses for the same part are tightly clustered. The discriminant filter in this case is the result of combining the Gaussian derivatives using those coefficients.

A survey of related approaches to texture-based image feature design is in Section 2, followed by the formulation of discriminant filters in Section 3. Part classification experiments which apply discriminant filters to three previously reported filter types are described in Section 4 and discussed in Section 5.

7.2 Previous Work

Our goal is to derive a feature mapping f which maps image patches P to a low-dimensional space. We will assume that the dimension m of the image feature space is selected ahead of time, and that the size of the image patches is fixed at k. We will also assume that we have access to training data, specifically example sets $\{(P_l, l)\}$ of image patches P_l corresponding to object parts l. Our approach to bottom-up localization begins by computing $f(P)$ for a given patch, and classifying the image feature $f(P)$ using a part classifier. In this section we review previous designs for the feature mapping and part classifier.

Several authors, including [22], [84], [104], and [56] propose the use of principal components analysis to model the local appearances of parts, much the way earlier researchers [80] [82] modeled global appearances. An eigenspace decomposition is computed for the set of all training patches (or Fourier transforms of them as in [56]), and at run time, image feature mapping consists of projecting novel image windows onto the first several significant eigenvectors. Since projecting patches into the eigenspace is a dot-product operation, the significant eigenvectors can equivalently be thought of as eigen-"filters" that are correlated with the test image patches. Local eigenspaces can be a good way to model the appearances of parts in a low-dimensional subspace since the first m principal components represent the best m-dimensional fit of all patches in an SSD sense; however for our application we are more interested in discriminating between sets of image patches than re-

constructing them. In particular, while eigenspace techniques maximize the covariance of filter outputs over all classes of image patches, they do not necessarily encourage partitions between them. As a result, it is necessary to tune parameters of the PCA decomposition– namely, the image patch size and number of significant eigenvectors– to ensure that eigen-filter responses for different object parts and for clutter are not confused. Reasonable settings depend on the scale of visual texture on the object and viewing conditions such as levels of occlusion, noise, clutter, and lighting.

A more prevalent approach to filter design is to construct banks of convolutional kernels based on criteria that do not depend on individual instances of training data. Gabor wavelets are an especially popular kernel choice [10] [81][102] since they are localized in frequency and space; it is easy to synthesize a set of Gabor kernels that regularly blankets spatial and frequency domains. Gaussian derivatives are also widely used [102][10][93] due in part to the fact that responses to them are equivariant to scale [102]; they have the added advantage that filter outputs for certain transformations of the image patches can be determined automatically by steering[38]. While these kernels form a mathematically sound way to represent the image signal present in the patches, responses from them are not necessarily sufficient for discrimination; in practice we will need to adjust the Gaussian widths of these filters, and the frequencies of Gabor kernels, to ensure that the information extracted from views of different object parts and clutter can be disambiguated from each other.

Invariants based on kernel responses are helpful since the outputs for the same object parts will not vary at all when the image of the part undergoes certain transformations; for example the differential invariants in "jet" space computed by Schmid *et al* [103] will not change if the image of the part undergoes a rigid displacement. Still, there is no guarantee that for a particular set of kernel parameters these invariants will be distinguishable for different parts. Again, to ensure discriminability, the settings of the kernel bank must be tuned.

Other image filters, for example those based on local contour invariants [106], color invariants[110], and Laplacian zero-crossing images [63] could suffer the same limitation– the parameters for these transformations may need fine-tuning to reduce confusion between outputs for distinct parts.

An expressive part classification mechanism may accommodate image filters whose outputs are not necessarily tuned for discrimination. Mohan, for example[81], gets good part classification results using Gabor kernel responses classified by a set of support vector

machines with nonlinear Mercer kernels, while Nelson *et al* [106] achieve high performance using contour invariants and a hash table. The drawback is that the part classifiers are governed by their own sets of parameters that must be tuned. In particular, the choice of Mercer kernel and penalty terms for SVMs affect false positive and false negative rates while bin size and policy for handing out votes are critical for indexing schemes to perform well. There is evidence that hashing schemes in particular are especially sensitive to parameter settings [47]. Worse, the effect of filter design and classifier design on performance is coupled– a change in the number of eigenvectors in PCA, for instance, may alter the design of k-nearest-neighbor distance functions which analyze the filter responses. Our results suggest that in some cases discriminant filter responses can cluster well enough that simple classifiers can perform well– for example, we see acceptable part classification results by fitting Gaussian distributions to outputs.

While discrimination-centered techniques have not been applied to image feature design in bottom-up recognition, they have appeared in single-step and top-down approaches, as well as in other domains such as texture discrimination. These related techniques are discussed in Section 2.1. Our approach is related to the Fisherfaces method[7], in which images of an entire object (faces in this case) are projected into a low-dimensional space using PCA, and a second linear transformation is determined by optimizing a Fisher ratio to encourage disparate outputs for different objects and similar outputs for the same object. Our approach differs in that we take many image transformations and combine their outputs, while Fisherfaces incorporate one eigenspace decomposition.

7.3 Approach

For notational simplicity we illustrate the approach for the case of discriminating between two parts with sets of example image patches $\mathcal{C}_1$ and $\mathcal{C}_2$. Our goal is to derive a feature mapping f which maximizes the following criterion:

$$\mathcal{R} = \frac{\frac{1}{|\mathcal{C}_1||\mathcal{C}_2|}\sum_{P_1 \in \mathcal{C}_1, P_2 \in \mathcal{C}_2} \|f(P_1) - f(P_2)\|^2}{\sum_{\mathcal{C}_p \in \{\mathcal{C}_1, \mathcal{C}_2\}} \binom{|\mathcal{C}_p|}{2} \sum_{P_1, P_2 \in \mathcal{C}_p} \|f(P_1) - f(P_2)\|^2} \tag{7.1}$$

The numerator summarizes the distances between projected patches in $\mathcal{C}_1$ and projected patches in $\mathcal{C}_2$ and is analogous to the between-class scatter of Fisher discriminants[25]. The denominator summarizes the distances between projected patches in the same set and is analogous to within-class scatter. We assume that two sets whose patches are well-

separated from each other will be the easiest to discriminate, so we seek a f which maximizes the numerator; at the same time, we assume that well-clustered sets of features require simpler representations for discrimination so we want f to minimize the denominator.

We express f as a linear combination of m-dimensional basis functions $\boldsymbol{f_j}$:

$$f(x) = \sum_j \alpha_j \cdot \boldsymbol{\phi_j}(\boldsymbol{x}), \quad \boldsymbol{\phi_j}(x) = \begin{bmatrix} \phi_{1j}(x) \\ \phi_{2j}(x) \\ \vdots \\ \phi_{mj}(x) \end{bmatrix}$$

This representation is not a restriction; we can represent arbitrarily complex functions f provided that we have a sufficient number of unique basis functions. Each $\boldsymbol{\phi_j}$ represents one set of filters for a particular parameter setting; returning to the Gabor kernel example, each $\boldsymbol{\phi_j}$ could correspond to a different choice of envelope width, with each ϕ_{ij} being a Gabor kernel with that width and some choice of orientation and frequency. Given a set of n basis functions $\boldsymbol{\phi_j}$ for n different parameter settings, we seek to find the set of coefficients $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_n]^T$ which maximizes $\mathcal{R}$. Substituting $\sum_j \alpha_j \cdot \boldsymbol{\phi_j}(\boldsymbol{x})$ for f in (7.1) and rearranging terms, we see that the numerator is equal to

$$\frac{1}{|\mathcal{C}_1||\mathcal{C}_2|} \sum_j \sum_k \alpha_j \alpha_k A_{1jk} - 2 \cdot \sum_j \sum_k \alpha_j \alpha_k B_{12jk}$$

$$+ \sum_j \sum_k \alpha_j \alpha_k A_{2jk}$$

and the denominator is

$$\binom{|\mathcal{C}_1|}{2} (2 \cdot \sum_j \sum_k \alpha_j \alpha_k A_{1jk} - 2 \cdot \sum_j \sum_k \alpha_j \alpha_k B_{11jk})$$
$$+ \binom{|\mathcal{C}_2|}{2} (2 \cdot \sum_j \sum_k \alpha_j \alpha_k A_{2jk} - 2 \cdot \sum_j \sum_k \alpha_j \alpha_k B_{22jk})$$

where

$$A_{pjk} = \sum_{i=1}^{m} \sum_{P_1 \in \mathcal{C}_p} \phi_{ij}(P_1) \phi_{ik}(P_1)$$

and

$$B_{pqjk} = \sum_{i=1}^{m} \sum_{P_1 \in \mathcal{C}_p, P_2 \in \mathcal{C}_q} \phi_{ij}(P_1)\phi_{ik}(P_2)$$

The ratio may be expressed equivalently as

$$\mathcal{R} = \frac{\boldsymbol{\alpha}^T N \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T D \boldsymbol{\alpha}} \tag{7.2}$$

where N and D are n-by-n matrices such that

$$N(j,k) = \frac{1}{|\mathcal{C}_1||\mathcal{C}_2|}(A_{1jk} - 2B_{12jk} + A_{2jk})$$

and

$$D(j,k) = \binom{|\mathcal{C}_1|}{2}(2A_{1jk} - 2B_{11jk}) + \binom{|\mathcal{C}_2|}{2}(2A_{2jk} - 2B_{22jk})$$

The $\boldsymbol{\alpha}$ which maximizes (7.2) is the eigenvector corresponding to the maximum generalized eigenvalue of N and D. Note that the coefficients A_{pjk} and B_{pqjk} which comprise N and D are readily computed by evaluating the basis functions $\boldsymbol{\phi_j}(\boldsymbol{x})$ over all elements of both sets $\mathcal{C}_1$ and $\mathcal{C}_2$ and taking various dot products and sums. The generalized eigenvalues of N and D may then be recovered using well-established numerical techniques.

This formulation is not restricted to two-part discrimination. In the general case, the within-class distances in the denominator will be summed over all sets and the across-class distances in the numerator will be summed over all pairs of sets, thus

$$N(j,k) = \sum_{\mathcal{C}_p \neq \mathcal{C}_q} \frac{1}{|\mathcal{C}_p||\mathcal{C}_q|}(A_{pjk} - 2B_{pqjk} + A_{qjk})$$

and

$$D(j,k) = \sum_{\mathcal{C}_p} \binom{|\mathcal{C}_p|}{2} 2A_{pjk} - 2B_{ppjk}$$

To summarize, our problem of combining basis functions to maximize distances across sets of image patches while minimizing distances within the sets reduces to evaluating the basis filters on the patches in the sets and finding generalized eigenvalues. As a concrete example, suppose we would like to use a vector of differential invariants based on derivatives of a Gaussian (as in [103]) for image features, but it is unclear how to choose one or more

Figure 7.2: A sample of cluttered scenes containing the mug.

Gaussian widths for the filters. We let each ϕ_j correspond to one such vector of invariants for a particular choice of σ; by varying σ discretely over a range we arrive at a set of ϕ_j functions which are combined using the derived coefficients α.

7.4 Experiments

We collected images of a common object (Figure 7.2) in varying poses and labeled the locations of selected object parts as in Figure 7.1(a). For each recognition experiment, we selected standard image filters from the literature and instantiated basis functions ϕ_j corresponding to a range of parameter settings for it. Given a subset of the labeled images as training data, we used the basis functions to derive discriminant filter coefficients as in Section 7.3, and used the remaining images for evaluation. To place the experiments in the context of previously reported end-to-end algorithms, we trained a nearest-neighbor part classifier as in [10][103][56][21][82][84][7]. For comparison, we also estimated Gaussian distributions for part classification as well. This section describes the data and experiments in detail.

7.4.1 Data

We took 60 images of a coffee mug with a hand-held camera. Of these photos, 12 featured the mug against a flat gray background and in the rest it was surrounded by a selection of

clutter objects (Figure 7.2). For each of these shots, the camera was at roughly the same distance and elevation from the object, but there were still slight variations in object scale and in all components of rotation since the camera positions were not carefully controlled. The clutter objects maintained the same spatial arrangement with respect to each other in 12 of the pictures; for the other 36 we moved the pieces around between frames. We labeled 18 parts on the mug in every view (Figure 7.1(a)). Each of them appeared in at least 20 images.

7.4.2 Experimental Procedure

For each filter basis and classifier, we ran trials consisting of the following steps:

- *Data Collection*

    1. For each part, randomly select 20 views of it and partition them so that 75% (15 views) are used for training, and testing is done on the remaining 25% (5 views).
    2. Select 100 image patches of clutter at random and partition these 75-25 into train and test sets.

- *Training*

    1. Solve the eigenvalue problem for discriminant filter coefficients over the parts and clutter training sets, treating the clutter as though it were another object "part."
    2. Store the discriminant filter responses for the training views and train a classifier based on them.
    3. Store the filter outputs for each ϕ_j on training patches and train a separate part classifier for the responses to each ϕ_j.
    4. Gather up all responses to each separate ϕ_j and train one classifier using all of them together as example data.

- *Testing*

    1. Run test set patches through each of the ϕ_j filters, compute the discriminant filters response from them, and process the results through each of the classifiers. In the case of the "all-ϕ_js-at-once" classifier, we follow a strategy found

in earlier approaches[103][102]: pick one ϕ_j and compute responses for it at run time.

We ran 25 trials of this sort for each classifier and filter basis. Specific characteristics of our classifiers are described next.

7.4.3 Classifiers

We represent our set of Nc part classifiers for Nc object parts as functions $\{c_1, c_2, \cdots c_{Nc}\}$. To train, we optimize these functions so that c_l is high for images of part l. At run time we gather up all scores $\{c_1(P), c_2(P), \cdots c_{Nc}(P)\}$ for all test image patches P and count how many correct class scores are above a certain threshold versus how many incorrect scores are above the same threshold. In other words, this assessment is somewhat "pessimistic" as one test example can account for many false alarms.

For k-nearest-neighbors, we compute the image feature $f(()P)$ for each test image patch P and find the k training image patches $\{P_{l1} \cdots P_{lk}\}$ from each class l whose image features $\{f(P_{l1}) \cdots f(P_{lk})\}$ are closest to $f(P)$. The class score $c_P(l)$ for P belonging to class l is then

$$c_P(l) = \sum_k exp(-C * \|f(P_{lk}) - f(P)\|^2)$$

We estimate the parameter C for each class by brute force at training time; for each P_l in class l, we compute $c_{P_l}(l) - max_{l_2 \neq l}(c_{P_l}(l_2))$ for 50 different settings of C ranging from 10^{-6} to 10^6. In the end we pick the C for which the median of these values is highest. We emphasize that while this optimization is time-consuming, it is exactly the sort of optimization that in some cases a nearest-neighbor classifier may require to ensure good performance; there are no theoretical reasons for the exponential in c_l to take on one value or another. In all experiments we set k to 5.

To compare performance with a less-flexible, more-easily-trained part classifier, we ran trials in which we fit a Gaussian distribution to training image features. In other words, the classifier for each part l is a Gaussian function:

$$c_P(l) = (1/(2\pi)^{d/2}\|\Sigma\|^d)exp((-1/2)(f(P) - \mu)^T\Sigma^{-1}(f(P) - \mu))$$

and at training time we use filter responses to estimate the mean μ and covariance Σ.

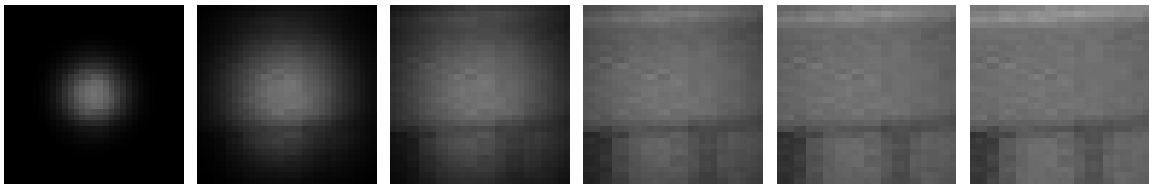


Figure 7.3: Image patch modulated by a range of Gaussian envelopes. For each Gaussian, a PCA decomposition is computed on the set of all training patches modulated by that Gaussian.

Since the covariance matrices Σ are estimated using a small number of examples relative to the dimension of the filters, we restrict Σ to be diagonal as in covariance selection methods [49].

7.4.4 Local Eigenfilters

The first set of experiments employs local eigenspaces for the image features. As in [22] [84][56], we want to collect all training patches for the various object parts and perform principal components analysis on them, but we would also like to automatically determine how to incorporate multiple patch sizes into our features. To apply discriminant filters to this problem, we simulate smaller effective window sizes by multiplying the patches by a Gaussian envelope. Depending on the standard deviation of this Gaussian, more or less of the periphery of the patch is set close to zero (Figure 7.3). We perform PCA on the set of Gaussian-modulated image patches at training time; at run time, a test patch is multiplied by the same Gaussian and projected onto the first few significant eigenvectors.

We assign each $\boldsymbol{\phi_j}(\boldsymbol{x})$ to a different width of Gaussian envelope, so that each $\boldsymbol{\phi_j}(\boldsymbol{x})$ corresponds to PCA on a different effective window size. More formally, let n_σ denote a Gaussian with zero mean and standard deviation σ, and let the set of all image patches be $\{\mathbf{p}_1, \mathbf{p}_2, \cdots\}$. If we write $\{\mathbf{v}_{\sigma 1}, \cdots, \mathbf{v}_{\sigma n}\}$ for the first n principal components of $\{n_\sigma \circ \mathbf{p}_1, n_\sigma \circ \mathbf{p}_2, \cdots\}$, then we set $\phi_{ij}(P) = \mathbf{v}_{\sigma_j i} \cdot n_{\sigma_j} \circ P$. There are 10 different $\boldsymbol{\phi_j}$ filters, ranging from $\sigma = .25$ to $\sigma = 2.5$. Each has 10 principal components, resulting in a 10x10 basis.

Results using the Gaussian classifier for discriminant filters are plotted solid on Figure 7.4; results where responses are extracted using individual $\boldsymbol{\phi_j}$ filters are shown dotted. Figure 7.5 shows the same plot for the nearest-neighbor classifier. Comparing plots on Figure 7.5 to each other, we see that discriminant filters with a k-nearest neighbor classifier can be competitive with previous local eigenspace techniques which use k-nearest-neighbors, with the advantage that multiple window sizes were incorporated automatically.

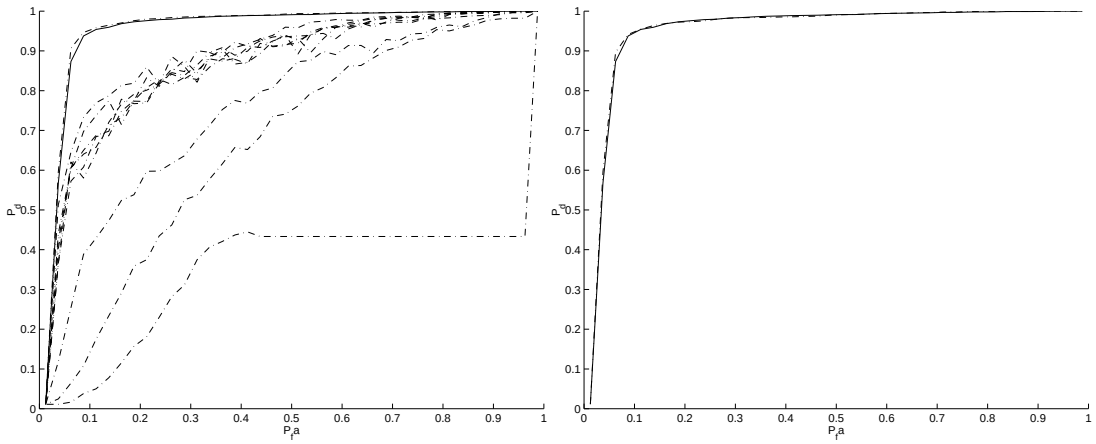


Figure 7.4: ROC curves using local eigenspace image features and a Gaussian part classifier. Solid curves on the left and right show performance of discriminant filters using eigenfilters as a basis. On the left, one dotted curve is plotted for each particular patch size. On the right, features for all patch sizes are combined at training time, and features for the median patch size are used at run time. Details in the text.

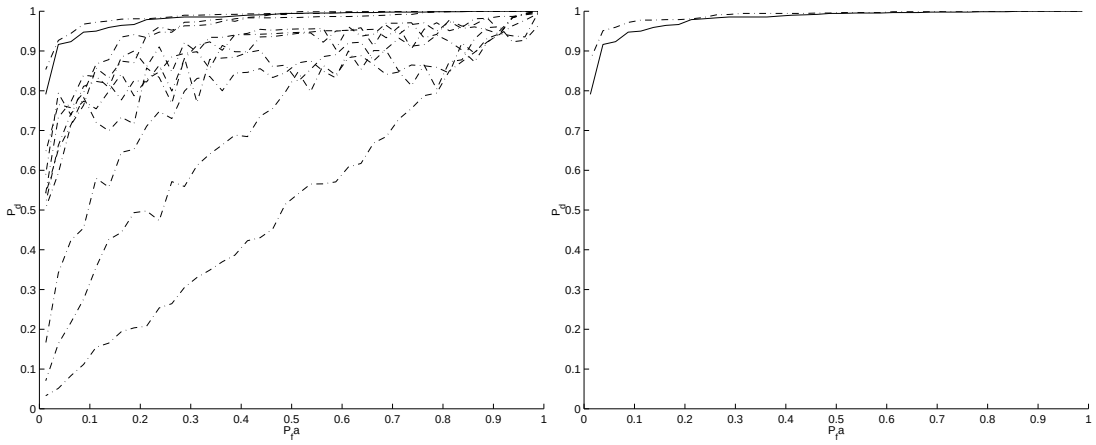


Figure 7.5: ROC curves using local eigenspace features and a k-nearest-neighbor classifier, as in Figure 7.4. Details in the text.

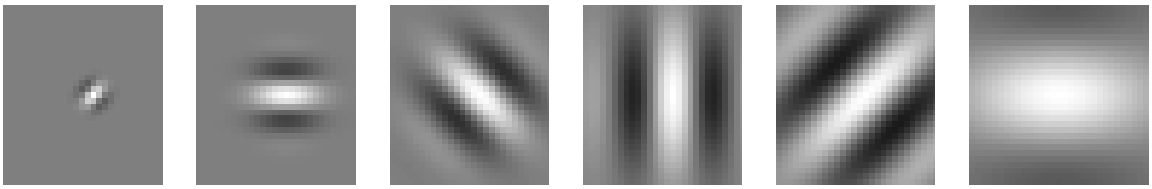


Figure 7.6: Examples of filters from the Gabor basis, varying by width of Gaussian envelope, frequency, and orientation.

Comparing Figure 7.5 to Figure 7.4 suggests that the use of a Gaussian classifier does not degrade performance, even though its parameters are much easier to estimate than those of k-nearest-neighbors. Furthermore, discriminant filters performance is almost identical to that of classifiers trained on all patches of multiple sizes.

7.4.5 Gabor Filters

Next we consider banks of Gabor filters, used in a number of recognition methods[81][10][102]. Gabor filters are sinusoids modulated by a Gaussian envelope; as above, we would like to use discriminant filters to determine what frequencies and Gaussian widths lend themselves to effective parts classification.

For these experiments, each ϕ_j consisted of a set of 4 Gabor filters oriented at even intervals between 0 and π radians. The 25 different ϕ_js correspond to each possible combination of 5 frequencies ranging evenly from .2 to .5 and 5 Gaussian widths varying from .1 to .75(Figure 7.6). All filters have a 1:1 aspect ratio. As above, we used discriminant filters to derive 4-dimensional image features over the 4x25 basis, and trained Gaussian and k-nearest-neighbor classifiers to discriminate them for the 18 object parts and clutter. Results are shown in Figure 7.7 and Figure 7.8 (left). While discriminant filters do not perform as well as the best single filter set with a Gaussian classifier, the performance is comparable, and discriminant filters achieve better part classification than storing responses at multiple scales at training time. As in the previous section, the key point is that we were able to compute the discriminant filters directly, rather than running recognition experiments for each parameter setting in turn.

7.4.6 Differential Invariants

The next set of experiments is applied to differential invariants in "jet" space [103]. To compute an nth-order differential invariant for an image patch, we convolve it with all

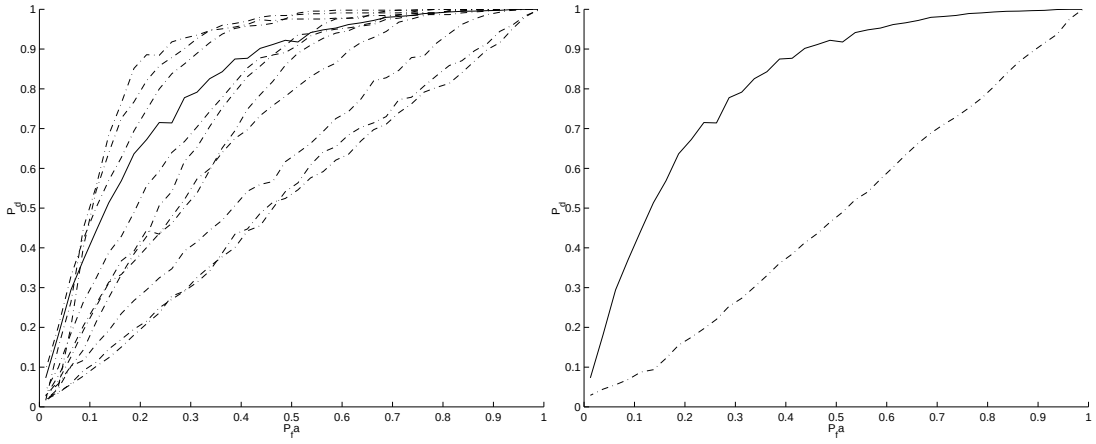


Figure 7.7: Part classification ROC curves using Gabor filters and a Gaussian classifier, displayed as in Figure 7.4. Details in the text.

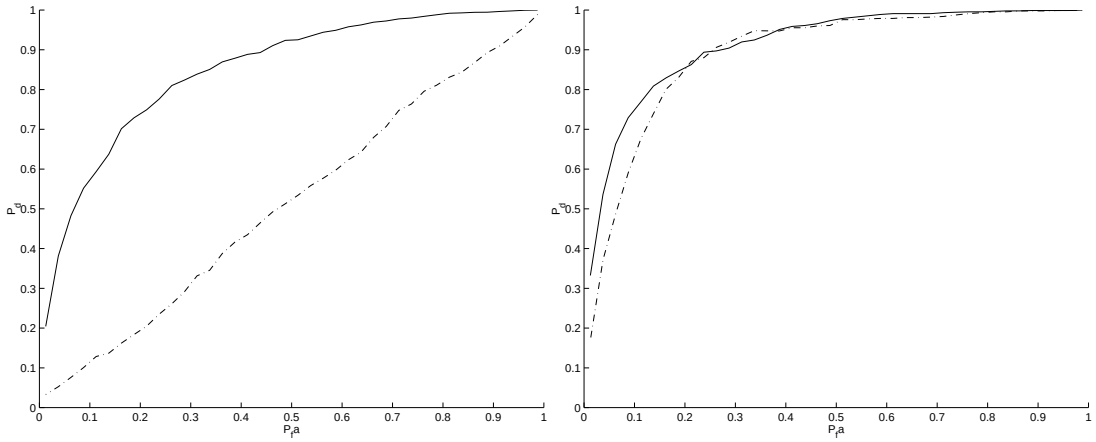


Figure 7.8: Left: ROC curves using Gabor filters and a nearest-neighbor classifier. The solid curve plots performance using discriminant filters; the dotted curve trains on all filter responses for all ϕ_j. Right: ROC curves using differential invariants and nearest-neighbor classifier, displayed as the graph on the left.

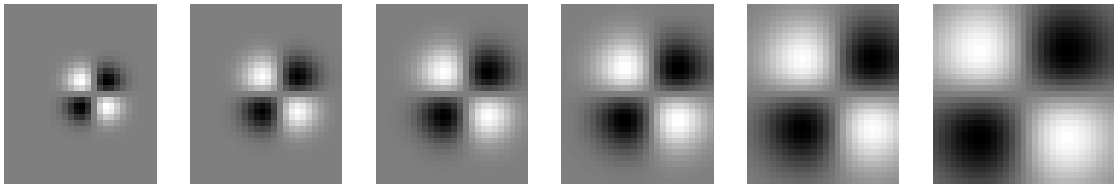


Figure 7.9: A filter basis is constructed using differential invariants computed from derivatives of Gaussians over a range of Gaussian variances. Shown is $\partial a / \partial x \partial y$ over that range of variances.

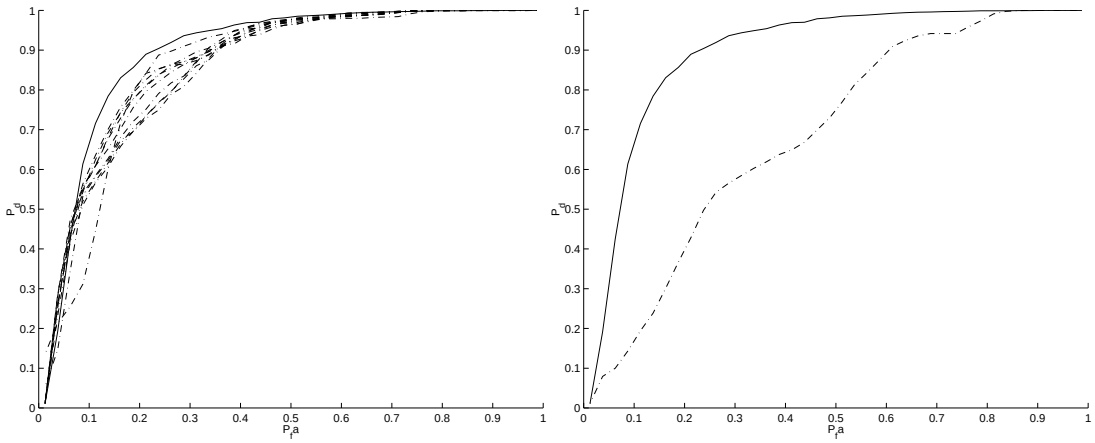


Figure 7.10: ROC curves using local differential invariant features and a Gaussian classifier. Solid curves show performance of discriminant filters using differential invariants of varying width as a basis. Left: One Dotted curve is plotted for each width setting. Right: Features for all widths are combined at training time, and features for the median width are used at run time. Details in the text.

Gaussian derivatives up to order n and construct invariants by multiplying and adding the results together. As in [103], our experiments focus on the use of 3rd-order differential invariants under the rigid displacement group; there are 9 such unique invariants, so each ϕ_j will be 9-dimensional.

These invariants are computed using derivatives of a single Gaussian, so we immediately arrive at the problem of determining what its standard deviation should be. In [103], Schmid *et al* compute the invariants over a range of discrete scales at training time and at a single scale at run time; here, we apply discriminant filters to the problem of selecting σ so that each patch is represented by a single vector of outputs during training. To do so, we select a set of values of σ and assign each ϕ_j to compute the differential invariants for a particular σ. We picked 10 values of σ ranging from .15 to .5, giving us a 10x9 filter basis (Figure 7.9). As above, we performed 25 trials using discriminant filters and 25 trials each for the individual settings of σ, using a Gaussian classifier. Figure 7.10 shows that in this case discriminant filters perform as well as the best setting of σ. Training a nearest-neighbor part classifier using all invariants computed for all scales, and at run time using the invariants for the median scale, gives results that are comparable to those for discriminant filters (Figure 7.8,right). They are also comparable to those for the Gaussian classifier, suggesting again that it is possible to achieve acceptable part classification by combining invariants at different widths automatically, without training a classifier on responses for all possible widths separately.

7.5 Future Work

In this section we address some of the computational and representational limitations of discriminant filters which should be addressed to broaden their applicability.

We present discriminant filters as a mechanism for optimizing image features based on labeled training image patches. Unfortunately, labeling the locations of a set of individual object parts in each image is time-consuming and tedious, especially if the number of object parts is high. Moreover, through hand-labeling object parts at training time, we assume that we have chosen a set of object parts that are salient for recognition. It could be the case that selecting which object parts are the most easily discriminable from each other and from the background depends on the choice of image features; by forcing the user to determine which object parts are salient *a priori*, we allow the user to design flaws into the recognition system. Future work should concentrate on automatic methods for determining which object parts are salient for recognition, and extracting those parts from images in which the spatial extent of the entire object is the sole piece of label information.

Our experiments show that in some cases, discriminant filters can obviate the need for fine-tuning the parameters of a bank of image features. Unfortunately, discriminant filters imply an additional computational burden at run time: rather than extracting a single bank of features from the image, we are forced to extract a set of feature banks corresponding to distinct feature bank parameters. It is true that in the case that each of the basis filters ϕ_j consist strictly of linear convolutions with the image patch– as in Sections 7.4.4 and 7.4.5– we are able to use the derived α coefficients to linearly combine the ϕ_j ahead of time so that at run time we only need to convolve the image with a single bank of m filters. However, if the ϕ_j functions contain nonlinearities– as in Section 7.4.6– then we will need to evaluate each ϕ_j at each image patch, which can be a computational drain for large numbers of ϕ_j. Another avenue for future research is biasing the optimization for the vector of combining coefficients $\boldsymbol{\alpha}$ toward sparseness, in other words encouraging as many of the entries in $\boldsymbol{\alpha}$. It is possible that sparsity-encouraging optimization techniques employed by the machine learning community for learning sparse kernel machines, for example [112][35], can be applied to the optimization of sparse $\boldsymbol{\alpha}$.

7.6 Summary and Conclusions

This chapter presents an approach to texture-based image features based on tuning sets of image filters. Given an arbitrary set of basis filters and sets of labeled patches containing the parts, we derive the combining coefficients needed to maximize discrimination between the filter outputs for the various classes. Initial part classification results on real data and image filters commonly used in the recognition literature support the validity of this technique. As opposed to previous approaches, this paper suggests that it is possible to derive useful image information from a linear combination of sets of basis filters which span a plausible range of parameter settings, rather than selecting one setting or storing responses to all possible filters separately. The filters in turn can enable simple classification mechanisms that do not require much tuning.

Chapter 8

Conclusions

The theme of this thesis is that solving object recognition as a discrimination problem makes it easier for us to expand the state of the art in the field and apply existing computer vision tools to new scenarios.

Our developments in shape-based localization show that discriminative techniques can facilitate shape-based recognition to scenes with busy, confusing backgrounds. To our knowledge, none of the shape-based algorithms presented thus far in the field are able to localize objects like the ladder, stool, chair, or cart in such complex images as the ones we experiment on. Previously presented techniques have serious difficulty in designing rich, informative edge features which are able to differentiate the object from the clutter; here, by tuning our edge-based features so that they directly discriminate between object and background, we provide a solution to that problem. Moreover, prior shape-based techniques have difficulties in setting parameters on their part classification and aggregation modules; we provide a solution which sets these parameters automatically by focusing on how the parameter settings lead to discrimination between object and clutter. Finally, the computational architecture of our technique leads to efficient performance at training and run-time.

Our contribution in texture-based image features illustrate that discriminative techniques can make the application of existing computer vision tools to object recognition simpler. The design parameters of banks of image convolutional image filters can be difficult to set by hand for recognition tasks; therefore, we provide a solution which combines sets of filters so that they discriminate well between object portions of the image and background portions. The design parameters of the classifiers which categorize filter responses

can be difficult to set as well; here, we show that discriminative image features can ameliorate this problem in some settings.

As we describe in Chapters 5 and 7, the main limitations to our approach are related to computational complexity and overfitting. With regard to overfitting, the problem is that we want to have it both ways– we want the simplicity of being able to optimize object recognizers so that they discriminate specific object images from specific clutter images, but we also want our recognizers to perform well on arbitrary, novel images. We see in the experiments in Chapter 5 that if the divergence between training and test images is large, recognition performance suffers. However, in Chapter 3 we see examples in which moderate divergence leads to only a modest reduction in performance. Our experiments involved a relatively modest number of training images (on the order of hundreds); we are confident that with a large amount of training data, our discriminative techniques for shape- and texture-based localization will generalize well to a range of novel images.

However, the idea of large amounts of training data brings us to our second limitation, which is computational complexity. First, as discussed in Section 6.3.3, we need to provide tools for our algorithms to efficiently sift through billions of training examples representing pixels in tens of thousands of images to pick which examples are unique and informative. And second, as discussed in Section 7.5, our approaches assume that the training data is hand-labeled, so we need to formulate algorithms for quickly labeling training images or incorporating semi-labeled or unlabeled images.

Addressing these limitations will help us to apply the algorithms of this thesis to larger image sets, more complex scenes and objects, and more sources of image variability. It is our belief that the framework we present here takes an important step toward the overall problem of recognizing realistic objects in real images of ever-increasing complexity.

Bibliography

[1] Hussein Almuallim and Thomas G. Dietterich. Learning with many irrelevant features. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1991.

[2] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers, 1997.

[3] Yali Amit, Donald Geman, and Bruno Jedynak. Efficient focusing and face detection. Technical Report 459, Department Of Statistics, University of Chicago, October 1997.

[4] J.L. Barron, D.J. Fleet, and S.D. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[5] Ronen Basri and David Jacobs. Projective alignment with regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(5):519–527, May 2001.

[6] Jeffrey Beis and David Lowe. Indexing without invariants in 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1000–1015, 1999.

[7] P. Belheumer, J. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces:recognition using class specific linear projection. In *Proceedings European Conference On Computer Vision*, 1996.

[8] P. Belhumeur and D. Kriegman. What is the set of images of an object under all possible illumination conditions? *International Journal of Computer Vision*, 28(3):245–260, 1998.

[9] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002.

[10] J. Ben-Arie, Z. Wang, and R. Rao. Iconic recognition with affine-invariant spectral signatures. In *Proceedings IAPR-IEEE International Conference on Pattern Recognition*, volume 1, pages 672–676, 1996.

[11] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[12] Avrim Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[13] Yuri Boykov and Daniel P. Huttenlocher. A new bayesian framework for object recognition. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, pages 517–523, 1999.

[14] Jeffrey P. Bradford, Clayton Kunz, Ron Kohavi, Cliff Brunk, and Carla E. Brodley. Pruning decision trees with misclassification costs. In *Proceedings of the European Conference on Machine Learning*, pages 131–136, April 1998.

[15] A.P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[16] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International, Belmont, CA, 1984.

[17] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[18] Andreas Buja and Yung-Seop Lee. Data mining criteria for tree-based regression and classification. In *Proceedings KDD*, 2001.

[19] M. Burl, M. Weber, and P. Perona. A probablistic approach to object recognition using local photometry and global geometry. In *Proceedings European Conference On Computer Vision*, pages 628–641, 1998.

[20] Ingemar J. Cox, Matthew L. Miller, Thomas P. Minka, and Peter N. Yianilos. An optimized interaction strategy for bayesian relevance feedback. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, 1998.

[21] V. Colin de Verdiere and J. Crowley. Local appearance space for recognition of navigation landmarks. *Journal of Robotics and Autonomous Systems*, 1999. Special Issue.

[22] Vincent Colin de Verdiere and James L. Crowley. Visual recognition using local appearance. In *Proceedings European Conference On Computer Vision*, pages 640–654, 1998.

[23] A. Divakaran, K. Peker, R. Radhakrishnan, Z. Xiong, and R. Cabasson. Video summarization using mpeg-7 motion activity and audio descriptors. Technical Report TR2003-34, Mitsubishi Electric Research Laboratories, May 2003.

[24] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the International Conference on Machine Learning*, 1995.

[25] Richard Duda, Peter Hart, and David Stork. *Pattern Classification*. Wiley-Interscience, 2 edition, 2001.

[26] J. P. Egan. *Signal Detection Theory and ROC Analysis.* Series In Cognition and Perception. Academic Press, New York, 1975.

[27] Charles Elkan. The foundations of cost-sensitive learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2001.

[28] Scott Fahlman and Christian Lebiere. The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, Carnegie Mellon University School Of Computer Science, February 1990.

[29] Ahmed A. Faisal, Markus Fislage, Marc Pomplun, Robert Rae, and Helge Ritter. Observation of human eye movements to simulate visual exploration of complex scenes. Technical report, Universitt Bielfeld SFB 360, 1998.

[30] Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. AdaCost: misclassification cost-sensitive boosting. In *Proc. 16th International Conf. on Machine Learning*, pages 97–105. Morgan Kaufmann, San Francisco, CA, 1999.

[31] U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 800–805, 1989.

[32] S. A. Feig. Breast cancer screening: potential role of computer-aided detection (cad). *Technology In Cancer Research And Treatment*, 1(2):127–31, April 2002.

[33] Winfried Fellenz. A sequential model for attentive object selection. In *Proceedings of the IWK*, pages 109–116, 1994.

[34] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, volume 2, pages 264–271, 2003.

[35] Miguel Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1150, September 2003.

[36] D.A. Forsyth and M.M. Fleck. Body plans. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, pages 678–683, 1997.

[37] David Forsyth, John Haddon, and Sergey Ioffe. Finding objects by grouping primitives. In D. Forsyth, J. Mundy, V. Gesu, and R. Cipolla, editors, *Shape, Contour, and Grouping in Computer Vision*, pages 302–318. Springer-Verlag, 1999.

[38] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[39] W. T. Freeman, E.C. Pasztor, and O. Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 20(1):25–47, October 2000.

[40] W.T. Freeman. Exploiting the generic viewpoint assumption. *International Journal of Computer Vision*, 20(3):243–261, 1996.

[41] João Gama and Pavel Brazdil. Cascade generalization. *Machine Learning*, 41(3), 2000.

[42] Donald Geman and Bruno Jedynak. An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):1–14, 1996.

[43] Athinodoros Georghiades, Peter Belhumeur, and David Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, June 2001.

[44] G.J. Giefing, H. Janßen, and H. Mallot. A saccadic camera movement system for object recognition. In *Proceedings International Conference On Artificial Neural Networks*, pages 63–68, 1991.

[45] Luc J. Van Gool, Theo Moons, and Dorin Ungureanu. Affine/ photometric invariants for planar intensity patterns. In *Proceedings European Conference On Computer Vision*, volume 1, pages 642–651, 1996.

[46] W.E.L. Grimson. *Object recognition by computer : the role of geometric constraints*. MIT Press, 1990.

[47] W.E.L. Grimson and D. Huttenlocher. On the sensitivity of geometric hashing. In *Proceedings International Conference On Computer Vision*, pages 334–338, 1990.

[48] Y. Guermeur, A. Elisseeff, and H. Paugam-Moisy. A new multi-class svm based on a uniform convergence result. In *Proceedings IJCNN*, 2000.

[49] David Hand. *Construction and Assessment of Classification Rules*. Wiley, 1997.

[50] Parag Havaldar, Gerard G. Medioni, and Fridtjof Stein. Extraction of groups for recognition. In *Proceedings European Conference On Computer Vision*, pages 251–261, 1994.

[51] Kim HS, Ishikawa S, Ohtsuka Y, Shimizu H, Shinomiya T, and Viergever MA. Automatic scoliosis detection based on local centroids evaluation on moire topographic images of human backs. *IEEE Trans Med Imaging*, 20(12):1314–1320, December 2001.

[52] G. W. Humphreys, M. J. Riddoch, and P. T. Quinlan. Cascade processes in picture identification. *Cognitive Neuropsychology*, 5(1):67–103, 1988.

[53] ”Katshuki Ikeuchi, Berthold K. P. Horn, Shigemi Nagata, Tom Callahan, and Oded Feingold. Picking up an object from a pile of objects. In Brady and Paul, editors, *Proceedings of the First International Symposium on Robotics Research*, pages 139–166. MIT Press, 1983.

[54] J. Illingworth and J.V. Kittler. A survey of the hough transform. *CVGIP*, 44(1):87–116, October 1988.

[55] Andrew Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5), 1999.

[56] D. Jugessur and G. Dudek. Local appearance for robust object recognition. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, 2000.

[57] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:920–932, 1994.

[58] Daniel Keren, Margarita Osadchy, and Craig Gotsman. Antifaces: A novel, fast method for image detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(7):747–781, July 2001.

[59] K. Kira and L. A. Rendell. A practical approach to feature selection. In *Proceedings of the International Conference on Machine Learning*, pages 249–256, 1992.

[60] K.Mikolajczyk and C.Schmid. An affine invariant interest point detector. In *Proceedings European Conference On Computer Vision*, volume 1, pages 128–142, 2002.

[61] Ron Kohavi, Dan Sommerfield, and James Dougherty. Data mining using MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence*. IEEE Computer Society Press, 1996. `http://www.sgi.com/tech/mlc`.

[62] I. Kononenko and E. Simec. Induction of decision trees using relieff, 1995.

[63] John Krumm. Object detection with vector quantized binary features. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, pages 179–185, June 1997.

[64] Y. Lambdan and H. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Proceedings International Conference On Computer Vision*, pages 238–249, 1988.

[65] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[66] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. *Feature Grouping*, chapter Object recognition with gradient-based learning. Springer, 1999.

[67] S. Z. Li and J. Hornegger. A two-stage probabilistic approach for object recognition. In *Proceedings European Conference On Computer Vision*, 1998.

[68] Stan Li, Long Zhu, ZhenQiu Zhang, Andrew Blake, HongJiang Zhang, and Harry Shum. Statistical learning of multi-view face detection. In *Proceedings European Conference On Computer Vision*, 2002.

[69] Rainer Lienhart, Luhong Liang, and Alexander Kuranov. A detector tree of boosted classifiers for real-time object detection and tracking. In *IEEE Int. Conf. on Multimedia and Systems (ICME2003)*, July 2003.

[70] X. Liu, A. Srivastava, and K. Gallivan. Optimal linear representations of images for object recognition. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, 2003.

[71] David Lowe. Object recognition from local scale-invariant features. In *Proceedings International Conference On Computer Vision*, pages 1150–1157, 1999.

[72] David G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Boston, Mass., 1985.

[73] J. MacCormick and A. Blake. A probabilistic contour discriminant for object localization. In *Proceedings International Conference On Computer Vision*, 1998.

[74] John MacCormick and Andrew Blake. Spatial dependence in the observation of visual contours. In *Proceedings European Conference On Computer Vision*, pages 765–781, 1998.

[75] Shyjan Mahamud. *Discriminative Distance Measures for Object Detection*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, July 2002.

[76] Shyjan Mahamud, Martial Hebert, and Jianbo Shi. Object recognition using boosted discriminants.

[77] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using brightness and texture. In *Advances in Neural Information Processing Systems (NIPS*, December 2002.

[78] B. Mel. Seemore: Combining color, shape, and texture histogramming in a neurally-inspired approach to visual object recognition. *Neural Computation*, 9:777–804, 1997.

[79] T. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1 edition, 1997.

[80] B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. In *Proceedings International Conference On Computer Vision*, 1995.

[81] A. Mohan. Object detection in images by components. Technical Report AI Memo 1664, Massachusetts Institute of Technology, September 1999.

[82] H. Murase and Shree Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[83] Stepan Obdrzalek and Jiri Matas. Object recognition using local affine frames on distinguished regions. In Paul L. Rosin and David Marshall, editors, *Proceedings British Machine Vision Conference*, volume 1, pages 113–122. BMVA, 2002.

[84] K. Ohba and K. Ikeuchi. Detectability, uniqueness, and reliability of eigen windows for stable verification of partially occluded objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):1043–1048, 1997.

[85] Taylor P, Hajnal S, Dilhuydy MH, and Barreau B. Measuring image texture to separate ”difficult” from ”easy” mammograms. *British Journal Of Radiology*, 67(797):456–63, May 1994.

[86] Christopher Pack and Richard Born. Temporal dynamics of a neural solution to the aperture problem in visual area mt of macaque brain. *Nature*, 409:1040–1042, February 2001.

[87] Constantine P. Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Proceedings International Conference On Computer Vision*, 1998.

[88] Thompson PM, Hayashi KM, de Zubicaray G, Janke AL, Rose SE, Semple J, Herman D, Hong MS, Dittmer SS, Doddrell DM, and Toga AW. Dynamics of gray matter loss in alzheimer’s disease. *The Journal Of Neuroscience*, 23(3):994–1005, February 2003.

[89] Arthur Pope and David Lowe. Vista: A software environment for computer vision research. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, 1994.

[90] F.J. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the International Conference on Machine Learning*, pages 445–453, 1998.

[91] J.R. Quinlan. *C4.5 : programs for machine learning*. Morgan Kaufmann Publishers, 1993.

[92] T. Randen and J. Husøy. Texture segmentation using filters with optimized energy separation. *IEEE Transactions on Image Processing*, 8(4):571–582, 1999.

[93] R. Rao and D. Ballard. An active vision architecture based on iconic representations. *Artificial Intelligence*, 78(1-2):461–505, 1995.

[94] Rajesh Rao, Gregory Zelinsky, Mary Hayhoe, and Dana Ballard. Eye movements in visual cognition: A computational study. Technical Report 97.1, University of Rochester Department of Computer Science, March 1997.

[95] Photo Industry Reporter. Gretag debuts photopro consumables for photohub 208. *Photo Industry Reporter*, 9(9), 2001.

[96] T. Rikert, M. Jones, and P. Viola. A cluster-based statistical model for object detection. In *Proceedings International Conference On Computer Vision*, 1999.

[97] Ronald Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

[98] Henry A. Rowley, Shumeet Baluja, , and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, January 1998.

[99] Y. Dan Rubinstein and Trevor Hastie. Discriminative vs informative learning. In *Knowledge Discovery and Data Mining*, pages 49–53, 1997.

[100] F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *Proceedings International Conference On Computer Vision*, pages 636–643, July 2001.

[101] Robert Schapire. A brief introduction to boosting. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999.

[102] Bernt Schiele and James Crowley. Object recognition using multidimensional receptive field histograms. In *Proceedings European Conference On Computer Vision*, pages 610–619, 1996.

[103] C. Schmid and R. Mohr. Combining greyvalue invariants with local constraints for object recognition. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, 1996.

[104] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, 1998.

[105] Henry Schneiderman and Takeo Kanade. Object detection using the statistics of parts. *International Journal of Computer Vision*, 2002.

[106] A. Selinger and R. C. Nelson. A perceptual grouping hierarchy for appearance-based 3d object recognition. Technical Report 690, University of Rochester Computer Science Department, May 1998.

[107] M. E. Sereno. Modeling stages of motion processing in neural networks. In *Proceedings 9th Annual Conference of the Cognitive Science Society*, pages 405–416, 1987.

[108] Martin I. Sereno. Learning the solution to the aperture problem for pattern motion with a hebb rule. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems (NIPS*, pages 468–476, 1988.

[109] I. Shimshoni and J. Ponce. Probabilistic 3d object recognition. In *Proceedings International Conference On Computer Vision*, 1995.

[110] D. Slater and G. Healey. The illumination-invariant recognition of 3d objects using local color invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):206–210, 1996.

[111] Chris Stauffer and Eric Grimson. Similarity templates for detection and recognition. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, 2001.

[112] Michael Tipping. The relevance vector machine. In *Advances in Neural Information Processing Systems (NIPS*, 2000.

[113] Peter Turney. Types of cost in inductive concept learning. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning (WCSL at ICML-2000)*, Stanford University, California, 2000.

[114] Nuno Vasconcelos. Feature selection by maximum marginal diversity: optimality and implications for visual recognition. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, volume 1, pages 762–769, June 2003.

[115] Olga Veksler. Stereo correspondence with compact windows via minimum ratio cycle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1654–1660, December 2002.

[116] R. Veltkamp and M. Tanase. Content-based image retrieval systems: A survey. Technical Report UU-CS-2000-34, Universiteit Utrecht Institute of Information And Computing Sciences, 2000.

[117] Paul Viola and William M. Wells III. Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24(2):137–154, 1997.

[118] Paul Viola and Michael Jones. Robust real-time object detection. Technical Report CRL 2001/01, Compaq Cambridge Research Laboratory, 2001.

[119] Niblack W, Barber R, Equitz W, and Flickner M et al. The qbic project: querying images by content using color, texture, and shape. In *Proc SPIE (Storage and Retrieval for Image and Video Databases)*, volume 1908, pages 173–187, 1993.

[120] H. Wallach. The direction of motion of straight lines. In H. Wallach, editor, *On Perception*. Quadrangle, New York, 1976.

[121] Isaac Weiss and Manjit Ray. Model-based recognition of 3d objects from single images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):116–128, February 2001.

[122] T. Weldon and W. Higgins. Designing multiple gabor filters for multitexture image segmentation. *Optical Engineering*, 38(9):1478–1489, September 1999.

[123] J. Weston and C. Watkins. Support vector machines for multiclass pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, 1999.

[124] D.H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[125] J. Wu, J. M. Rehg, and M. D. Mullin. Learning a rare event detection cascade by direct feature selection. In *Advances in Neural Information Processing Systems (NIPS*, 2003.

[126] Stella X. Yu and Jianbo Shi. Object-specific figure-ground segregation. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, Madison, Wisconsin, June 2003. IEEE.

[127] Bianca Zadrozny and Charles Elkan. Learning and making decisions when costs and probabilities are both unknown. In *Knowledge Discovery and Data Mining*, pages 204–213, 2001.

[128] W. Zhao, R. Chellappa, A. Rosenfeld, and P. Phillips. Face recognition: A literature survey. Technical Report CAR-TR-948, University of Maryland CfAR, 2000.

[129] S.C. Zhu, R. Zhang, and Z. Tu. Integrating bottom-up/top-down for object recognition by data driven markov chain monte carlo. In *Proceedings IEEE Conference On Computer Vision And Pattern Recognition*, 2000.