StreetScenes: Towards Scene Understanding in Still Images

by

Stanley Michael Bileschi

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2006

[June 2006]
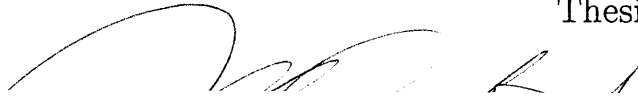
Author . . . .
Department of Electrical Engineering and Computer Science
May 5, 2006

Certified by. . . . .
Tomaso A. Poggio
McDermott Professor
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students
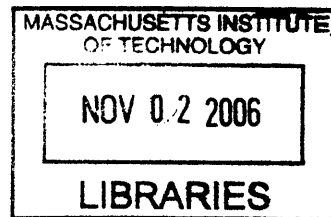
StreetScenes: Towards Scene Understanding in Still Images

by

Stanley Michael Bileschi

Submitted to the Department of Electrical Engineering and Computer Science
on May 5, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

This thesis describes an effort to construct a scene understanding system that is able to analyze the content of real images. While constructing the system we had to provide solutions to many of the fundamental questions that every student of object recognition deals with daily. These include the choice of data set, the choice of success measurement, the representation of the image content, the selection of inference engine, and the representation of the relations between objects.

The main test-bed for our system is the CBCL StreetScenes data base. It is a carefully labeled set of images, much larger than any similar data set available at the time it was collected. Each image in this data set was labeled for 9 common classes such as cars, pedestrians, roads and trees. Our system represents each image using a set of features that are based on a model of the human visual system constructed in our lab. We demonstrate that this biologically motivated image representation, along with its extensions, constitutes an effective representation for object detection, facilitating unprecedented levels of detection accuracy. Similarly to biological vision systems, our system uses hierarchical representations. We therefore explore the possible ways of combining information across the hierarchy into the final perception.

Our system is trained using standard machine learning machinery, which was first applied to computer vision in earlier work of Prof. Poggio and others. We demonstrate how the same standard methods can be used to model relations between objects in images as well, capturing context information. The resulting system detects and localizes, using a unified set of tools and image representations, compact objects such as cars, amorphous objects such as trees and roads, and the relations between objects within the scene. The same representation also excels in identifying objects in clutter without scanning the image.

Much of the work presented in the thesis was devoted to a rigorous comparison of our system to alternative object recognition systems. The results of these experiments support the effectiveness of simple feed-forward systems for the basic tasks involved in scene understanding. We make our results fully available to the public by publishing our code and data sets in hope that others may improve and extend our results.

Thesis Supervisor: Tomaso A. Poggio
Title: McDermott Professor

Dedication

To Mom and Dad.

Acknowledgments

Contents

List of Figures

15

17

19

22

List of Tables

Chapter 1

Introduction

Scene understanding, loosely, is the process of extracting meaning from visual input. When we say that a human *understands a scene*, we mean that the human could, with proper prior experience, answer questions about the scene regarding the presence and locations of objects, the identities or expressions of particular people in the scene, and, in video, the detection actions or events. We might even expect said human to be able to make conjectures about more complicated and abstract structure from the scene, producing inferences about social relationships or impending events in the world of the scene. The complicated and varied capabilities of the human scene understanding apparatus are certainly well beyond the most advanced computational methods of today.

In this thesis we will describe in detail the structure, implementation and properties of a system designed to take as input still images of street scenes, and output the detection of a diverse set of object categories, including objects like cars, buildings, people, and trees. This narrow, yet challenging scope was selected so as to provide focus to our efforts toward scene understanding as a whole. We will describe said system, compare the performance of its individual parts to the appropriate states of the art, and then, in later chapters, describe improvements and extensions to the system with a mind toward making the system more accurate in performance and more elegant in structure.

It should be noted that the resulting system, with proper training data, may be

applied to domains other than street scenes. There is no fundamental reason why it should not function if trained to recognize objects in other scenarios, such as office scenes or images of sea ports.

1.1 Goals and Motivations

There is no need to describe the motivation behind studying scene understanding as a whole. It is widely believed that the implications of the advent of robust automatic scene understanding would be dramatic in a wide variety of applications, including surveillance, robotics, health, and transportation, among others. Giving machines access to an abstract executive representation of their environment will enable a host of future technologies.

While meager compared to the abilities expressed by the human scene understanding apparatus, the focus selected for **this** thesis describes a fundamental, yet unsolved problem in scene understanding, i.e., how to reliably detect a wide variety of different object classes in a natural scenes. The goal of this work is then to build a test which closely measures performance on this sort of natural-scene object-detection problem, demonstrate aptitude at this test, describe our methods, and finally to share the test with the academic community so as to promote the healthy comparison of systems.

In order to manage the scope of this endeavor, we have decided to limit the domain to still images taken from the street and the detection of nine associated object categories. Still images, rather than video, was selected as the focus for this project for computational and database concerns. It may even be that the still image understanding problem is the more difficult in that there are no motion cues to assist in segmentation or recognition. The chosen domain, and the associated objects, were selected because they exhibit many of the properties which make object detection a difficult task, i.e., strong within-class variation due to changes in form, lighting, pose, etc., and wide variability in the types of cues necessary to detect the specific object type. Also, intuitively, street-scenes, like many categories of visual stimuli, exhibit a type of regularity of structure which we might envision leveraging, e.g., that cars

28

tend to drive upon roads. Finally, the understanding of street scenes is notable for the direct ties to important applications in safety and surveillance.

It is true that object detection in still images has been a focus of computer vision research for many years, and a short overview is given in section 1.2. The technology has come very far since its inception, as demonstrated by the year-by-year performance improvement on publicly available databases, such as those described in [71, 82, 105, 78, 64] and many notable others. In this light, it is fair to ask what is missing from these previous studies which is addressed in this thesis.[1] Again, our goal is the accurate detection of a *wide variety* of objects in a *natural environment*. Systems measured using the aforementioned databases and associated measures are in a way divorced from this goal, either by specialization to one particular object type, or by testing without clutter, where the approximate scale and position of the target object are already known. Furthermore, experiments are often performed on clean image crops with statistics well removed from the distribution found in natural images. It is often difficult to generalize from these studies how well the systems described therein would perform on a real world detection task. This is not to say that previous work in object detection has been ill conceived or poorly tested. Our position is simply that the field has matured to the point where it makes sense to build and compare systems in more difficult scenarios which better mimic the real world problem we are trying to address. In this work we will address these issues by providing a common natural-scene database and appropriate performance measures for object detection systems, describing a system which detects these objects, and comparing its performance to the state of the art systems in the field.

Again, there is certainly more to scene understanding than can be described in terms of object detection. It has been shown, for instance in Torralba *et al.*[119], that some certain scene-understanding type questions, such as depth-of-scene or generalized location, may be answered without any object-detection at all. However, many questions about the nature of the constituent objects and their relationship to the viewer may not be addressed without first detecting them within the scene. It is likely

---

[1] A more detailed account of the academic contributions of this thesis is given in section 1.4.

that any future technology that solves scene-understanding will have object detection as a component.

1.2   Overview of Previous Work

In this section we will discuss prior work in the areas of object detection, texture recognition, and context awareness. These technologies are all particularly relevant to our scene understanding task. In this short survey of related works we will focus on the data structures and learning techniques used. Subsequently, we will discuss the types of databases and performance measures commonly used today by the computer vision community.

1.2.1   Object Detection

In his thesis, Schneiderman [100] imagines an "ideal but infeasible" classifier for object detection. This classifier consists of a table in which every row corresponds to one image, discretized into say 16 bit grayscale 20 × 20 pixel images. Associated with each row is whether this image is an object or a non-object. As this table would have to have more than 10^{900} rows in order to represent every possible image, it certainly is infeasible. Even more perplexing, however, is that, as illustrated in [119], the same image, in different contexts, may be perceived in completely different ways, suggesting that even the table would not suffice to solve the object detection problem. In this section, the difficulties inherent to the object detection problem are discussed, followed by a short survey of current state of the art approaches.

A robust detector needs to compensate not only for changes in the position, scale, and pose of the object, but also for variability in lighting, possible occlusions, and especially for the changes in the particular example of an object class. It would be of little use to have to train a new classifier for every example of the object class. Figure 1-1 illustrates some selected examples of objects in natural scenes undergoing these sorts of transformations. Perhaps the most difficult types of objects to detect are those for which their nature is defined by their suitability to a certain function,

as opposed to their appearance or structure. Objects such as chairs, and perhaps vehicles, fall into this category.

When we speak of the object detection problem, there is an ambiguity that needs to be resolved. Often times, the term object-detection also includes the concept of object-localization, where the object's position and size are part of the answer. In contrast, object-detection may also refer to the binary problem of whether an image does, or does not contain an example of the target object, independent of where it is or how big it is within the image. In this thesis we will often refer to the later problem as "object-detection in clutter", and to the former problem as simply object detection. When the training set contains enough variability in the position and scale of the positive object class, we refer to it as object recognition using an unsegmented training set. Modern object detection systems have exhibited some proficiency in learning with these types of databases, but still underperform those systems with access to training databases with greater correspondence between examples. See, for example, [129, 38, 70, 103, 28, 7].

While early computer vision systems used model-based approaches [62, 45], nearly all modern computer object detection systems rely upon statistical learning methods. This requires that the images themselves must be represented mathematically, usually as some vector in $\Re^n$. As an example, the most naive way to represent an image is to simply resize the image to a standard size, convert it to grayscale, and record an ordered list of all the pixel values. It has been established, however, that detector performance depends critically on the choice of representation [52]. A wide variety of image representations have been used in the past, including grayscale pixel values, wavelet coefficients, linear projections (i.e., PCA, LDA and variants), and many others. Each has particular strengths and weaknesses for certain object types and modes of invariance. Recently, with the emphasis on performance, there has been a rapid convergence of representational choices into a few groups, described in more detail below.

Patch-Based Features. A patch-based feature vector is an image descriptor which depends on comparing the image with a set of stored image crops (also known

Figure 1-1: Large variations in appearance can occur within each object class. Changes in lighting and pose, varying amounts of occlusion, and individual object variability contribute to the difficulty of the object detection task. From top to bottom, this figure illustrates examples of the *bicycle, tree, car, building,* and *pedestrian* classes.

32

as templates or fragments). Common examples include those of Ullman and Sali [121], Torralba *et al.*[117], and Leibe and Scheile [68]. Different implementations strike different balances between invariance and the representation of geometric structure. It has been shown that patch based approaches can significantly improve performance over detection via the use of a single template, but they are still limited in representative power by the underlying measure used to compare the image with the stored prototype. Common choices include correlation, normalized cross-correlation, and Euclidean distance. Further advancements have replaced grayscale patch matching with matching in richer representations, such as a local histogram of edge orientations representation.

Histograms of Edge Orientations. This increasingly popular type of representation has demonstrated discriminative power for many types of objects and tolerance for several common image transformations. It can be described as a weighted histogram wherein each histogram bin collects votes from pixels near predefined image locations and at particular gradient orientations. Examples include Freeman *et al.*'s early work [40], Lowe's SIFT [74], Riesenhuber and Poggio's C1 features from their Standard Model [93], Berg and Malik's geometric blur [8], Belongie *et al.*'s shape context [6], and Dalal and Triggs' HoG [29].

The SIFT feature, described in detail in [74], was designed to be used in concert with an interest operator in order to find correspondences in images of the *same* object under different lighting and view angles. It therefore possesses several properties that make it suboptimal for *generic* object recognition. For example, it does not use rectified orientations, i.e., up and down are two different orientations. Also, it is normalized by the dominant orientation in the image. It was noted in [74] that the SIFT feature is a general framework, and its parameters can be modified to suit individual tasks. This was done in [29], where the *Histogram of Gradients feature* (HoG) was introduced and shown to be much more suitable for generic object recognition.

In the HoG, a histogram is defined wherein each bin represents a spatial region as well as a particular orientation range. In the published work, the best results

33

were obtained with one bin per 8 pixels in either spatial direction, and four or more orientations uniformly distributed from $0° - 180°$ (using rectified orientations). In order to calculate the feature vector, the magnitudes of the brightness gradient at each pixel are added to the appropriate histogram bins using linear interpolation between the nearest eight bin centers in location and orientation. Finally, a normalization step is applied. The performance of the HoG feature, for the task of pedestrian detection, over a wide variety of parameters, histogram structures, and normalization techniques, is explored in [29]. In our experiments we use the version with the optimal set of parameters.

The C1 feature [93, 102] stems from Riesenhuber and Poggio's Standard Model of biological vision, which will be discussed in greater detail in chapter 3. The C1 feature is very similar in structure to the edge histogram, but the smooth summation which is used to divide votes from edge detections into histogram bins is instead replaced with a local maximum operator. Each C1 element is the maximum edge response within a local neighborhood of position, scale, and orientation space.

Bag of Words Feature Many modern object detection systems begin with an interest operator *i.e.*, a function which, given an image, returns a number of points in scale space which are in some way mathematically interesting. Common examples include the Difference of Gaussian, Förstner, and the Harris-Laplace interest operators [75, 48, 79]. The benefit of this preprocessing is that the image is reliably and reproducibly dissolved into a sparse set of locations, which can then be each analyzed with greater scrutiny. The bag of words feature, in general, refers to collecting a sample of points from positive examples of the target class, and then representing a new image by how well each of these points is matched in the new image. This type of feature has been used in not only object detection [33, 111], but also in texture recognition [64, 108] and natural scene categorization [72]. It should be noted that the interest operator is not necessary for a bag of words feature, some systems sample uniformly across the image. For instance, Grauman *et al.*[44] sample features using a uniform grid, and match to the training set using a specialized *pyramid match kernel* to achieve state of the art performance on the Caltech 101 object recognition

benchmark.

Part Based Features Also known as a component based representation, this is a method of representing an object in terms of the confidence in detection of object-specific parts. Early work, such as [69, 86, 80] used hand selected parts which seem naturally salient to humans, such as the eyes, nose, and mouth for faces, or the head, shoulders, and legs for pedestrians. Other systems have been designed afterwards to learn object parts automatically from training data [95, 121, 52, 111, 70, 2]. The intuition behind part based representation is that while the object is susceptible to wide variations in appearance due to lighting, occlusion, and pose variations, the parts of the object are less vulnerable. While it has been shown that part based representations have enormous potential for accurate detectors, there are several weaknesses to the method. The first problem is deciding which parts to use and automatically constructing a training set for them, in order to build a classifier. Secondly, many objects are not well described by a geometric arrangement of parts. For instance, roads have a reliable appearance, but are not easily expressed in terms of parts. Finally, the detection of the parts themselves is another object detection task, and is a recursion of the problem we are trying to solve in the first place. The detection of the parts is usually performed using one of the other features mentioned.

Features Specifically for Texture Recognition In this thesis we will be detecting some objects, such as roads and skies, which are more recognizable by their textures than by their shape or organization of constituent parts. Although, there is some overlap between object detection and texture recognition. Features commonly used to represent textures include Haralick co-occurence matricies [91], features based on the statistics of the local distribution of filters [2, 19, 106], Malik's Texton feature [92], and the bag of words feature [64]. In the system described in this work we will employ the C2 feature from Riesenhuber and Poggio's Standard Model [93, 102] in a texture recognition sense. This will be described in more detail in chapter 3, where will compare methods of texture based object detection.

Representation is only one piece of the puzzle when designing an object detection system. Another equally important factor is how the representation will be used. A

major division is drawn between generative and discriminative systems. Generative systems model the object and the non-object classes probabilistically and then output the more likely category a posteriori. Non-parametric systems, like k-Nearest Neighbors, are often used for their simplicity, but are limited in their empirical performance. Parametric generative object detection systems model the object and non-object distributions using some hand-selected probability distribution. The parameters are usually set automatically using some form of parameter fitting, such as Expectation Maximization (EM) algorithms [49].

Discriminative systems, instead of modeling the distributions, attempt to divide the representation space into object and non-object zones. The most commonly used methods are support vector machines (SVMs), boosting, and regularized least-squares classification [49, 94]. In the systems described in this thesis, due to the (in general) large number of features and small number of examples, we will rely most heavily on boosting and linear-kernel SVMs. The implementations used in this thesis are gentleBoost [49], and Chang *et al.*'s LIBSVM [25].

1.2.2 Context Awareness

The features and methods described above are concerned with the principled detection of objects via their **appearance**. It is well known however, through psychophysical and biological experiments [10, 27, 4], that object perception in humans depends additionally upon image features outside of the object. These features are referred to as the object's **context**. Without getting too tangled in semantic issues, this visual context might be defined as *those image features which are relevant to the object detection task, but not affected by the object presence.* Previous work has used many types of contextual features to aide in object detection, such as the nature of nearby objects [81, 110, 46, 23], the relative position and scale of those objects [107, 60, 39, 9, 47, 13], as well as statistics of low level visual features of the scene as a whole [115]. In general, in natural images, objects are strongly expected to fit into a certain relationship with the scene, and context gives access to that relationship.

Previous context-enabled systems may be grouped into three sets: systems which

share information via a network of object-detectors, systems which classify the scene as a whole, and systems which employ a graphical model over a segmentation of the image. As an example of a system from the first set, Torralba et al. [117] employ boosting and graphical networks to learn associations between the likely co-occurrence and relative position of objects within a scene. Fink describes a similar system [39] for which detections of objects' parts, as well as detections of other objects in a scene, are employed in a modification of the Viola-Jones cascaded object detection architecture [126]. This type of dense composition leaves no information source untapped, but the downside of such structures is that any mutual dependencies must be computed in an iterative fashion, first updating one object then the other. Moreover, if the target object is the only labeled object in the database then there are no sources of contextual information. Systems which pre-segment the image and then model the relationships between neighboring segments, i.e. [61, 23], suffer from similar issues.

Mutual dependance is not a problem for systems which use context by processing the scene as a whole, without first detecting other objects. Murphy et al.[116] employ context as a scene 'gist', which influences priors of object existence and global location within that scene. The disadvantage here is that the scene must be taken as one complete unit and spatially localized processing can not take place.

Some researchers believe that context only makes sense in a generative framework, using for example random fields or graphical models of context and focusing on the expected spatial relationships between objects. Even assuming that the world is best described by such a hierarchal generative process, there is no reason to believe that accurate and useful classifiers can not be built using a discriminative framework. This is one of the main results of Vapnik's statistical learning theory [125].

In this thesis we describe a new discriminative system for context recognition which is simple to implement and feed-forward. It uses the relative positions of other detected objects in the scene as well as low-level cues such as global positions, colors and textures to build a map of the contextual support for the target object. The internals of this algorithm are detailed in Chapter 4.

37

1.2.3 Databases and Performance Measures

Modern object detection systems require training databases. Similarly, in order to measure the performance of the system, we require a testing database. It has become very important for the vision community to have publicly available databases and performance measures so as to fairly compare and contrast the qualities of different methods. The COIL, CMU PIE, Roth's car detection, and more recently the Caltech 101 Object databases [82, 105, 2, 71], among many notable others, have provided opportunities to compare systems between labs and demonstrate improvement within the field. It is worthwhile to discuss a bit further exactly what sorts of measurements are used to judge object detection systems, so as to understand the benefits and shortfalls of the various options.

Restricting the discussion to binary object detection (as opposed multi-class detection) for the moment, there are at least three distinct ways, excluding variations upon these themes, of measuring the performance of an object detection system. Multiclass detection measures are a bit less straightforward, and typically involve averaging performance measurements across many binary problems.

Probably the most common measure of object detection performance is to build a test database of image crops wherein each crop either contains an example of the target object, or does not. Depending on how much variability there is in the position and scale of the target object, this can be considered as a detection in clutter task, as described in section 1.2.1. Examples of the use of this sort of measure abound [30, 70, 35, 123, 66]. We will refer to this as the *crop-wise* measure, as it describes the process of pulling crops of objects and non object from a database of natural images, and then classifying the crops individually. The advantage of this sort of measure is in its simplicity. With a bit of care, a test database can be built for which each image either contains an example of the target object, or it does not; there is not much room for ambiguity. Bootstrapping can be used to collect more difficult test examples, if the test is too easy. The disadvantage of this sort of measure is that, as noted by Roth in [2], because of the differing sampling strategies of the positive and negative

test crops, the statistics of the test crops do not match very well with the statistics of uniformly random crops taken from natural scenes. As a result, the measure is more reflective of the classification power of the machine than its ability to reliably detect objects. This can also be considered an advantage of this method in that the classification power can be measured apart from concerns of different windowing or neighborhood-suppression algorithms. Another confounding issue in the crop based measure is the question of exactly how to fairly crop the examples from the scenes. As classifiers can rely upon more or less of the surrounding context, there can be an inherent bias in the manner of cropping. When presenting the results of the crop-wise measurement, in the binary case, performance is illustrated via the receiver operator characteristic (ROC), or, equivalently, Detection Error Tradeoff (DET) [30] curve. If a single dimensional measurement value is required, then often the area under the ROC curve, the equal error rate, or the true positive at some hand selected false positive rate is used.

Another, perhaps more natural measure of performance requires that objects be detected in full scenes. The bounding box of each detection is compared to a set of ground truth boxes, which are often hand drawn. In order to determine the true positives and false detections, some measure of box overlap is used. This measure of performance, which we will refer to as the *box-wise* measure, has been used recently in the PASCAL object detection challenge[2], as well as a large number of past object detection works (particularly in face detection) [100, 89, 87, 2]. The advantage of measuring the detector performance box-wise is that it simulates the process of actually detecting the object in natural scenes. The types and distributions of true positives and false positives encountered in the scenes should match those of an actual application. The disadvantage lies in the ambiguities in what constitutes an accurate positive detection and the additional complexity involved in windowing and neighborhood suppression, or whatever technology is used to generate candidate locations. In box-wise detection, it doesn't make sense to use the ROC curve, since the number of negative windows in the data set is undefined. Instead, precision-recall (PR) curves

---

[2]http://www.pascal-network.org/

are used to measure performance.

When dealing with amorphous objects such as roads or buildings, the above measures are inappropriate. There is too much ambiguity in which rectangular box best outlines the object. In this case, two strategies are possible. One is to only take crops from *within* the object, instead of around it, and measure performance as in a *crop-wise* problem. Another strategy is to treat each pixel as an independent classification problem, using the human labeling as a ground-truth mask. After the classifier has operated on the pixels, true positives and false positives are judged via ROC. An extension of this method has been used by Barnard in [23], where the image is first segmented, and then segments are classified individually. One undesirable property of these methods is that objects which contain more pixels (or equivalently are divided into more segments) have more weight on the final measure. In our texture recognition experiments we use this measure and will refer to it as the *pixel-wise* detection measure.

A fourth distinct measure of object detection performance is similar to the crop-wise measure, except that the crops are not separated from the image as a whole. Instead, each positive and negative test example is given in terms of a full image and a region of interest. This region of interest can be defined as a bounding box, polygon, location and scale, *etc.* The task here is to tell whether this region represents the target object or not. The critical difference between this measure and the crop-wise measure is that detection systems have access to the whole image, from which contextual information may be drawn. The difference between this and the box-wise measure of performance is that the detection system is not responsible for whatever mechanism generates candidate locations for classification, reducing the computational complexity of the task while maintaining the utility of the comparison.

In order to have an appropriate training and testing database for **our** scene understanding studies, we have collected a large set of digital camera images taken in and around the city of Boston. Of this database of nearly $8,000$ images, $3,547$ of them have been hand labeled with the location of nine object categories. This database will be discussed in more detail in Chapter 2. Our database and implementations of the

associated performance measures have been made publicly available on the internet [11].

1.3 Thesis Outline

This thesis describes a system for scene understanding in natural images and several extensions thereto. We will begin the next chapter by discussing our image database and appropriate object detection measures. In chapter 3, the structure of the full scene understanding system is described, including several baseline measures of performance. Afterwards, the thesis consists of extensions upon this basic scene understanding model.

Chapter 4 discusses a novel system for discriminative contextual modulation wherein object detections can influence the detection of other objects, leveraging the natural structure of the scenes. This context classifier is combined with the detectors designed in chapter 3 to improve detection scores.

More extensions are described in chapter 5, focusing on novel image features. It will be shown here that features designed to reproduce gestalt-like qualities assist in the detection of real world objects. These new features are compared in efficiency and power to similar randomly generated image features. We will also discuss in this section a powerful language for describing image features and the feasibility of searching the space of image features for highly discriminative subsets.

In chapter 6 we will discuss some research into feedback mechanisms for hierarchies of visual features. We will demonstrate that depending on the task, different structures are more or less successful.

Chapters 7 and 8 discuss the scope of the thesis, what has been accomplished, and what areas are most ripe for further discovery. We predict fruitful future directions within this line of research in this section.

1.4 Main Contributions

- The main contribution of this thesis is the Street-Scenes infrastructure [14]. The detailed description of the system is written in chapter 3. A system has been built which is capable of detecting a wide variety of object categories, from buildings to trees to pedestrians, in natural scenes. Furthermore, careful measurements indicate that the individual components of this system are at least on par with the state of the art. These performance levels are further improved in chapters 4 through 6. The system as a whole is more an art of engineering than it is of science, as much of what comprises the system, such as the standard model features, texture recognition, windowing and local neighborhood suppression, has been documented in previous publications.

- A contextual learning system, built on top of our previous architecture, is described in chapter 4. This context leveraging system is different from previous attempts at context aware systems and demonstrates that effective contextual biasing can be performed in a discriminative framework [130].

- In chapter 5 we describe improvements through the development of novel image features. These features are an attempt to capture mid-level visual regularities, such as continuity, closed contours, symmetry and parallelism. A biologically motivated algorithm for calculating each is presented, and the benefits of using such features are measured, such as in the Caltech 101 object database where these features have enabled state-of-the-art levels of performance [16].

- Another contribution not to be overlooked is the public StreetScenes database and performance measures. The database and its properties are described in chapter 2. It is hoped that this public measure of scene understanding will further meaningful task-oriented comparisons between systems in the vision community [12].

- Finally, we offer an ontology of feedback mechanisms for combining feature modalities in visual hierarchies [131]. These feedback mechanisms have enabled

42

us to further improve performance in our object detection mechanisms.

Chapter 2

The StreetScenes Database

The street scenes database is a collection of images, annotations, and performance measurements designed to train, test, and quantify the performance of an object detection system. In this chapter we will discuss in detail how this database was constructed, what advantages and limitations it has over other object detection databases, and what choices were made during the implementation of the detection performance measures.

2.1 Why Build Our Own Database

While there exist object detection databases that involve the detection of multiple objects, such as the COIL and Caltech 101 database [82, 71], among many others, and there are also object detection databases which involve the detection of objects in real scenes, such as the car detection database of Roth [2], until recently there has not been a database which involves the detection of multiple image categories in natural scenes. Our goal in this project was to learn to dissect entire **scenes**, giving meaning to a majority of the pixels in each image, paving the way towards even deeper scene understanding. StreetScenes is the first publicly available database of a large number of labeled images from the same scenario where each image is labeled consistently with the same object categories and using the same criteria. As there are a wide variety of different objects to detect, it is possible to use the database to explore contextual

notions and the relationships between different objects. We selected scenes taken from the street as the domain for this project for several reasons, including the variety of objects which reliably appear, the regularity of the structure within the scenes, and the obvious applications to security and surveillance of a system which understands images of public areas.

A selection of StreetScenes images and their associated hand labels are illustrated in Appendix A.

2.2 Building the Database

2.2.1 Image Acquisition

The collection of the StreetScenes images was performed by payed employees of the Center for Biological and Computational Learning (CBCL) during the period between January 2003 and July 2004. The images were captured using two identical Sony DSC-F717 Digital cameras, using the automatic settings, and setting the capture resolution to 1280×960 pixels[1]. The camera was set to automatically compress the images into JPG format, and they were transferred via IP to a secure FTP site, where they would later be labeled. Photographers (of which there were a total of 13 people) were instructed not to interfere with the camera's default settings.

Before being allowed to capture photographs, photographers were given hands-on instruction on the type of pictures and procedures necessary for the project. Photographers were instructed to take pictures on plane with the horizon, tilting the camera neither toward the ground nor toward the sky. Pictures were to be taken from the sidewalk, cross walk, or the street, and subject matter was to be selected as randomly as possible from the 180° arc from straight along the road in one direction to along the road in the reverse direction. Photographers were instructed not to take photographs which were too similar. This was to be accomplished by not shooting the same subject matter without moving about 30 feet first. All photographs were

---

[1]The camera is capable of a maximum resolution of 2560 × 1920 pixels.

	number of examples	mean width	mean height	mean center (x)	mean center (y)
car	5,799	344	189	635	531
pedestrian	1,449	89	196	640	530
bicycle	209	179	199	628	591
building	5067	590	461	635	290
tree	4,932	413	382	627	275
road	2,562	518	249	642	135
sky	3,400	1,167	460	643	720
sidewalk	3,658	742	265	629	720
store	590	401	281	623	397

Table 2.1: Some basic statistics of the labeled objects in the StreetScenes database. All measurements are given in pixels. For reference, each image is 960 × 1280 pixels, and there are a total of 3,547 labeled images in the database.

taken between dawn and dusk, there are no night-shots, and there are no photographs taken in the rain, although there is fallen snow visible in some of the images.

Photographers were given subway fare and instructed not to densely photograph locations which have already been recorded. By the end of the project, nearly 8,000 StreetScene images had been collected from various areas such as urban Boston and suburban Summerville. Some images were removed due to inappropriate location, subject matter, lighting, motion blur, or bad camera settings.

2.2.2   Image Labeling

Of the database of StreetScenes images, exactly 3,547 have been hand labeled. In order to hand-label an image, a human must draw a polygon around each example of any of the nine selected object categories. The selected categories, and some statistics, are described in table 2.1. Close individual instruction was given to labelers of the database so as to promote consistency throughout the database. Inevitably, however, there is some variability in how the polygons were drawn. Some examples of object labels are illustrated in appendix A. More detail into the labeling process is given on an object-by-object basis below.

cars In the StreetScenes database, the car class includes all motorized vehicles

with more than two wheels, including cars, trucks, busses, and construction vehicles. Cars are labeled only if they are larger than approximately 64 pixels in width and more than 75% visible, or, equivalently, less than one quarter occluded. Labelers were instructed draw the polygon as tightly as practical around the car, imagining the continuation of the car if it were occluded. This choice was made so as to improve correspondence between examples normalized in position and scale. Cars, and other objects, which exit the scene are sometimes labeled with polygons which exit the frame of the photograph. The photograph boundary was considered to be an occluding boundary.

pedestrians Similarly to cars, pedestrians are labeled only if they are more than 75% visible and larger than 32 pixels in height. The database contains pedestrians walking as well as standing still. In some situations where many pedestrians are visible in a group the labelers were instructed to do their best to label the pedestrians which were the least occluded.

bicycles Bicycles are a difficult class with comparatively few examples in the database. Labelers were told to label bicycles with and without riders. Motorcycles were also included in the bicycle class. Bicycle racks containing too many bicycles to label individually are labeled with one bicycle polygon, as if bicycles were a texture.

buildings The building class includes structures such as skyscrapers, office buildings, and domestic homes. Because it is often difficult to determine where one building ends and the next begins, multiple buildings are very often included in one bounding polygon. The labelers were instructed to imagine the continuation of a building if it is obscured by cars, trees, or the like.

trees The tree class includes summer foliage as well as barren winter trees. It was impractical to label each tree individually, so multiple trees are usually included in one bounding polygon. Because of the sparse nature of the trees in winter, labeling with polygons is particularly difficult. Labelers were instructed to capture as much of the tree as possible within the polygon. It was noticed that labelers sometimes imagine tree-trunks with their polygons even when none are plainly visible. This practice was discouraged.

48

roads The vast majority of road examples are paved, although there exist a few examples of construction work which are labeled as road. Labelers were instructed to imagine the boundary of the road heading into the vanishing point, and it was to be labeled even if partially occluded by cars or pedestrians. The road was not labeled when it was occluded by buildings or completely occluded due to heavy traffic.

skies The sky class is labeled wherever open sky is visible. Ambiguous cases occur where trees gradually grow dense enough to render the sky invisible. Labelers were to use their own judgement as to where the sky ends.

sidewalks The sidewalk class was labeled using rules similar to the road class. The sidewalk boundary is imagined to be continuous even when the actual boundary is obscured by pedestrians or parked cars.

stores The store class is probably the most ambiguous class labeled in the StreetScenes database. Labelers were told to draw polygons around the store fronts of buildings with "signs" or "where they sell things."

All labeling of the StreetScene database was performed with a custom image labeling tool implemented in MATLAB. Fig. 2-1 presents a screen shot of the user interface of this tool running in a windows version of MATLAB. This image labeling tool allows a user to browse directories for images, open labeling files, view previously made annotations, and add or delete annotations. In general, the photographers mentioned in sec. 2.2.1 were the same individuals who were employed in the labeling process. On average, images were labeled at the rate of one per 3 minutes.

It should be noted that a very similar methodology for labeling images was developed by Torralba *et al.*in the design of the LabelMe database [96]. LabelMe is another image database, similar to StreetScenes, but the topic is not similarly constrained to one domain. The LabelMe database is an open database, in that users are allowed to log in and add images and labels via an internet application. Between the two databases, there are a number of differences. StreetScene images are constrained to be only images of the street, taken with the same camera, and the same nine objects are labeled in the same way, as far as practical, in each image. LabelMe has a larger database with a much wider variety of objects and more variability in the labeling

49
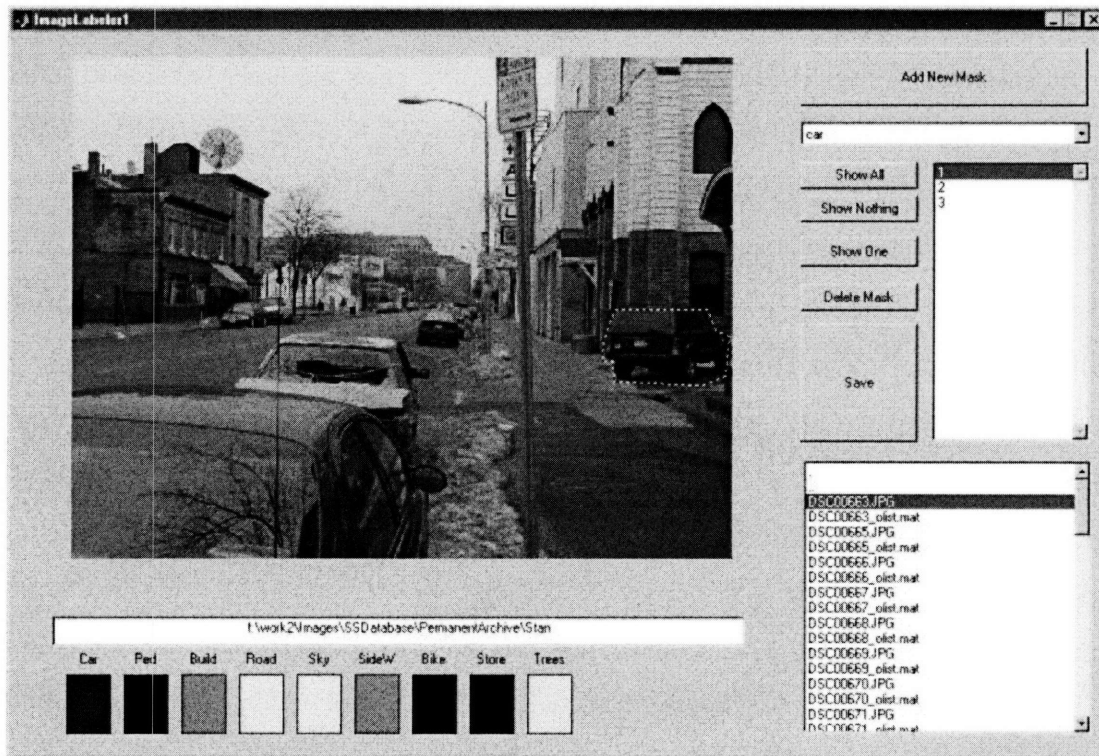
Figure 2-1: A screen-shot of the interface to our MATLAB image labeling tool.

policy. Translation has been supplied along with the StreetScenes database so that the two databases are fully compatible.

2.3   Performance Measures

Along with the images and the annotations, the StreetScenes database includes a package of MATLAB functions for working with the database and measuring object detection performance. As mentioned previously, there are a number of unique ways of measuring the performance of an object detection algorithm, and because different performance measures are more or less applicable to certain object types or types of detector, we have implemented the four detection measures outlined here. More detailed information on the use of the associated MATLAB functions, including interfacing and parameter structure, is available in appendix B

Crop-wise detection measure These functions allow the user to extract positive and negative crop examples of target objects from the database. Negative samples are selected from a distribution of scales that matches the distribution of the positive examples. Negative samples are allowed to overlap the labeling of a positive object example by no more than 10 percent. The crops are then randomly split into training and testing sets, and multiple trials are performed. The same random splits are repeatable, allowing for greater statistical significance when comparing performance between classification systems. The measure outputs the ROC curve and associated statistics.

Box-wise detection measure The box-wise detection measure compares a list of proposed object detections, in terms of their bounding boxes and associated confidences, to the bounding boxes of the baseline hand-drawn polygons. A detection is considered to be a true positive if it matches well enough to a baseline bounding box. Without getting into too much detail, the detection A is considered to be a match to a baseline object B if $\frac{area(A \cap B)}{area(A \cup B)} > \theta$ where θ is a free parameter controlling how close A and B have to be to match. The default value, used in our experiments, is $\theta = \frac{1}{2}$, and is the same as used in the PASCAL object detection challenge. One difference

from the PASCAL challenge is that in the challenge, objects can have variable aspect ratios, *i.e.*, they can be short and wide or thin and tall. The implication is that in order to achieve the θ overlap criterion, some crude level of object segmentation has to take place. In our system there is an option to mandate that all bounding boxes will adjust to have the same square aspect ratio, making the task slightly less segmentation oriented. The measure outputs the PR curve and a datastructure indicating which detections matched which baseline objects.

Pixel-wise detection measure Conceptually, the pixel-wise detection measure is very simple, a number of image locations are selected and must be labeled. In the implementation we provide, these points are automatically selected by extracting a certain number per target example. A system parameter controls how close to the polygonal border these points may be selected, defaulting to 15 pixels. Due to the imperfections in polygonal labeling is important to set this parameter wide enough that there is confidence that most points lie well within the object. The algorithm controls that points from the same original image are either all training or all testing. In other words, if two points are extracted from the same image, they will either both be training, or both be testing.

Box-wise detection measure with provided boxes This is a measure very similar to the box-wise detection measure, but candidate boxes are provided by the test. In the binary case, positive boxes around known object locations are provided along with similarly distributed boxes of known negative locations. The task is for the classifier to determine which boxes stem from positive locations. It is possible to generalize to the multiclass case by simply providing boxes sampled from each desired class. This test simulates an attentional mechanism where object candidates need to be verified and labeled. This task also has the property that the entire image is visible to the detection mechanism and can be leveraged for context unlike in the crop-wise measure.

52

2.4 Limitations of the Database

The StreetScenes database is a powerful tool for training and testing scene-understanding algorithms. There are, however, several shortcomings and limitations to the database.

When hand labeling the objects it was found that there is a great deal of ambiguity as to when an object should be labeled, and when it should not. For instance, in some photographs taken in the densely urban area during rush-hour, the entire road is invisible due to the dense traffic. Furthermore, each car is mostly occluded by the car behind it. As a result, under the rules defined above, neither the road nor the cars can be labeled at all. A future addition or improvement to the database may include 'car-as-texture' as an object type, wherein any part of a car or cars which is visible can be labeled. Other objects have similar ambiguous labeling situations. Also, due to time limitations and labeler fatigue, many objects are not labeled as tightly as they could be by the bounding polygon.

2.5 Download and Use

The StreetScenes database is available for download and use at the web-page for the Center for Biological and Computational learning[2]. The terms and conditions of the use of the database are also available there. The database includes images, annotations, and code for measuring performance.

---

[2]http://cbcl.mit.edu

Chapter 3

StreetScene Understanding Using the Standard Model Features

This chapter describes the design and implementation choices made in the development of our object detection system. As will become apparent, different types of object require different strategies for detection, but every detector within this system relies upon the same image features, *i.e.*, the *Standard Model Features* (SMF) derived from Poggio *et al.*'s Standard Model of biological vision [93, 103]. This model, and the associated features, will be described more completely in the next section, 3.1. The classifiers and detection structure are described afterwards in sections 3.2 through 3.4. After reporting some measures of system performance, we will discuss the system in general terms and compare it to other state of the art vision systems.

3.1   The Standard Model Feature Set

The Standard Model is a quantitative theory modeling the early stages of the feed-forward path of object recognition in primate visual cortex. It is so called because it attempts to summarize a core of well-accepted facts about this system: that early visual processing builds invariance through a hierarchy of processing stages, that the receptive fields of units along the hierarchy increase in size as does the complexity of the preferred stimulus, that the initial stages do not *require* feedback, and that there

is greater plasticity at higher visual stages than lower ones. For a comprehensive survey of the model and associated biological and psychophysical experimentation, please see the relevant literature [93, 103, 102]. It should be noted that the Standard Model describes a set of properties that such a vision system should have, but different model instantiations are possible. We will use the term "Standard Model" to refer to the model in abstract as well as the implementation used here.

In this thesis we will make use of this model by using the values at intermediate stages of one particular instantiation of the model as visual features. These features will be used as an image representation to our computer vision system. In section 3.1.1 we will describe the standard model and then afterwards elaborate how it relates to *computer* vision in section 3.1.2.

3.1.1 Introduction to the Standard Model

The architecture of the Standard Model, as it is instantiated for this work, is summarized in Fig. 3-1. This hierarchal model consists of four layers of computational units wherein *simple* (S) units alternate with *complex* (C) units. The S units combine their inputs using RBF-like operations, responding strongly only when their input is close (in an L_2 sense) to their preferred input. The C units respond with the maximum of their inputs, thereby introducing invariance.

The first S layer, layer S_1, consists of units with preferred responses in the form of oriented Gabor wavelets [42], as defined by equation 3.2. The appropriate parameters for these filters have been selected by matching biological neural responses from brain area V1. The receptive fields of these units are tesselated in such a way that they cover the scale-space of the image. The exact parameters used in this implementation are listed in Table 3.1, but in brief, 16 different filter sizes are used, and four different filter orientations. The S_1 units respond with the absolute value of the filter response.

The units of the next layer, C_1, pool inputs from the S_1 layer by responding with the maximum value of their input. Each C_1 unit is connected to a local scale-space neighborhood in S_1, *i.e.*, a small convex region in position and scale. The size of this neighborhood and the density of the tessellation are parameters of the model.

56
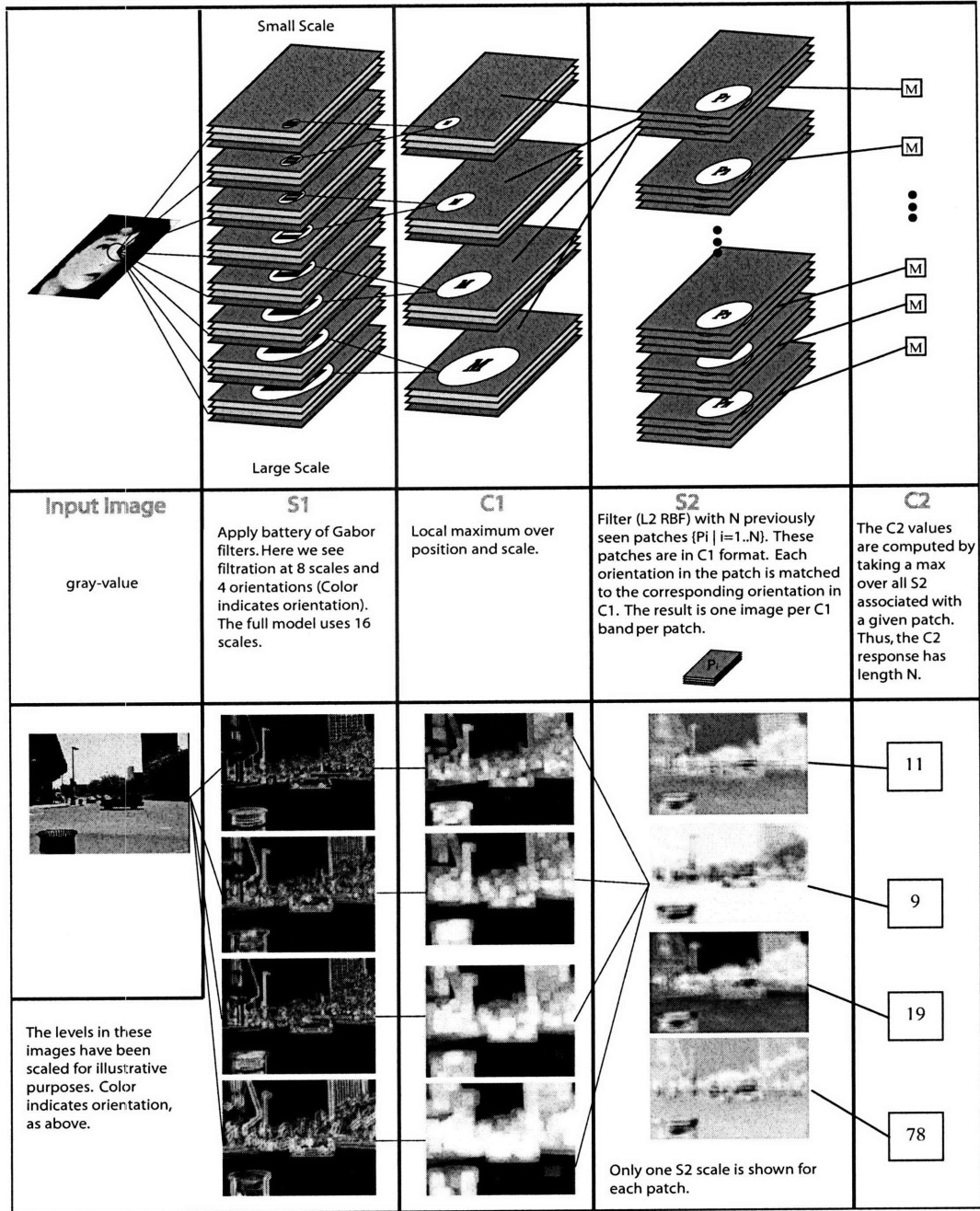
Figure 3-1: An illustration of the 4 hierarchal levels of the Standard Model system used in this work. The top half includes a schematic of the information flow through the system. The bottom half includes some real images sampled from the model as it processed an image from the StreetScenes database.

C_1 layer			S_1 layer		
Scale Band S	Spatial pooling area	sub-sample length	filter size	Gabor σ	Gabor λ
Band 1	8×8	4	7×7 9×9	2.8 3.6	3.5 4.6
Band 2	10×10	5	11×11 13×13	4.5 5.4	5.6 6.8
Band 3	12×12	6	15×15 17×17	6.3 7.3	7.9 9.1
Band 4	14×14	7	19×19 21×21	8.2 9.2	10.3 11.5
Band 5	16×16	8	23×23 25×25	10.2 11.3	12.7 14.1
Band 6	18×18	9	27×27 29×29	12.3 13.4	15.4 16.8
Band 7	20×20	10	31×31 33×33	14.6 15.8	18.2 19.7
Band 8	22×22	11	35×35 37×37	17.0 18.2	21.2 22.8

Table 3.1: The S_1 units perform Gabor filtering at many scales, and their responses are pooled over position and scale in by the C_1 units. This table is a summary of the S_1 and C_1 model parameters (see also Fig. 3-1 and accompanying text and equations for details).

In our implementation, C_1 units pool over two adjacent scales, and a spatial region dependant on the scale band. The C_1 units did not pool over orientation, as there were only 4 distinct orientations in the S_1 layer. C_1 units were tesselated such that the density was reduced in the scale direction and also in both spatial directions.

$$F(x,y) = \exp\left(-\frac{(x_o^2 + \gamma^2 y_o^2)}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda}x_o\right), \quad \text{s.t.} \tag{3.1}$$

$$x_o = x\cos\theta + y\sin\theta \quad \text{and} \quad y_o = -x\sin\theta + y\cos\theta. \tag{3.2}$$

In the S_2 layer, like the S_1 layer, units perform an RBF like function over their inputs. Whereas in the S_1 layer the input was the raw pixel values, and the preferred stimulus pattern was Gabor like patterns, in the S_2 layer the input is taken from units in the C_1 layer, and the preferred stimuli are a set of N patches $P \equiv \{P_i | i \in 1..N\}$. These patches are small crops of previously seen stimulations of the C_1 layer, selected by randomly sampling patches from natural images. Each patch includes information

from the full set of orientations, but only one scale band. If the patch has a spatial extent of 8 pixels, then it will consist of $8 \times 8 \times n_\theta$ values, where n_θ is the discrete number of orientations, in our case, 4.

Each S_2 unit is connected to a local neighborhood of C_1 values, *i.e.*, the C_1 values with receptive fields centering over the same small range of x, y, and orientation θ. The S_2 unit compares this vector of input X to P_i and computes an RBF (Eq. 3.3), where γ is a free parameter controlling the tightness of the tuning[1].

$$S_2 = \exp\left(-\gamma ||X - P_i||^2\right) \qquad (3.3)$$

The units in the S_2 layer are organized so as to tessellate the C_1 scale space. In addition, at each location there are exactly N S_2 units corresponding to the different P_i. In our implementation, patches were cropped at four different sizes: 4, 8, 12, and 16 pixels.

The C_2 units, similarly to the C_1 units, compute a maximum over their inputs. In some implementations of the Standard Model, this maximum is over the entire scale-space. This means the entire image is would be reduced to N values corresponding to the best match to the N patches. Newer, more detailed models involving higher processing add C_2 units such that, like the C_1 units, the maximum is computed only from a subset of S_2 units near to some location in the S_2 scale-space. In this case different C_2 units represent different positions and scales within the image. The size of this pooling region is another parameter.

3.1.2 Standard Model Features from a Computer Science Point of View

.

The first layer, S1, is an application of Gabor filters [42, 31] to the input image, which is fairly standard and has been used for many computer vision problems [19, 98, 83]. The S2 and C2 layers are an application of a patch based approach, which uses correlation with smaller image crops as its basic building block. In these types

---

[1]This free parameter also was set by matching biological data.

of systems, the best match to some memorized prototype is returned within some legal search area. As mentioned in section 1.2.1, this method has become common, and has been used successfully for texture synthesis [34], super resolution [41], object detection [122, 117] and object-specific image segmentation [18].

The only layer which might seem unorthodox from a computer vision perspective is the C1 layer, in which the outputs of the S1 layer are being maximized locally. While many systems maximize the output of a detector over the entire image, this has been done locally only recently. For part based object detection [122, 117], detectors of each part are learned independently and then applied to regions where the parts are expected to appear. The SMF seem unique in that *general purpose filters* are maximized over local regions in the image.

In order to explain the utility of C1, we invoke a scale-space terminology (see [73] for an overview). scale-space theory was mostly concerned at first with the Gaussian scale-space. This scale-space has many desirable properties such as separability, linearity, shift invariance, isotropy, homogeneity, and causality. The last property is an important one: causality means that no new level sets are generated by moving to coarser scales. A related property is the non-creation of local extrema in coarser scales.

In our application, local maximization is used to move from a fine scale to a coarser scale in order to make the C1 layer invariant to local edge translations. As a pseudo scale-space, local maximization has some desirable properties: it is separable (one can apply it over the rows and then over the columns), it is shift invariant, and it is homogeneous (applying it repeatedly corresponds to moving into coarser and coarser scales). However, in general, it is not an appropriate scale-space. Among other problems, applying it to an image may create new local extrema.

However, in the SMF framework, the local maximum operator is applied to a set of Gabor filtered images, which are known to be a *sparse* representation of the original image. The max scale-space is successful in preserving the amplitude of the sparse maxima, whereas the Gaussian scale-space smooths them out.

60

3.1.3 Comparisons to Other Commonly Used Vision Features

The vision features derived from the Standard Model have some properties which make them in some ways similar to features already in use by the computer vision community. As mentioned in chapter 1, one may wish to draw a comparison between the C_1 feature and the popular histograms of oriented edge filter features [74, 6, 29]. Both features collect local oriented edge responses and report them in a way which captures some local invariance to position and scale. The difference here being that in order to build a histogram, the edge responses are *added* into bins, *i.e.*, the bins are in a way taking a weighted sum of candidate edge locations. In contrast, the C_1 features are reporting the *local maximum* of the afferent S_1 oriented edge units.

The most striking difference in these two strategies is in the structure of the stimuli which provokes their strongest response. Since the C_1 unit is using a max, it's preferred stimulus is one in which there is at least one very strong edge response within its receptive field. In contrast, since histograms rely upon weighted sums, the stimulation which provokes the maximum response will be one in which there is a great deal of oriented energy at all locations within the histogram bin's receptive field. One reason why C_1 might be preferable in this situation is illustrated in figures 3-2 and 3-3. As edge responses in natural images tend to exhibit characteristics of sparsity [84], it may be very useful to be able to distinguish sharp local signals from low-level spatially-extended ones, such as from textures.

The C_2 features have a parallel within the field of patch-based features since, in essence, each S_2 feature is computing an RBF against a preferred patch. This patch, rather than grayscale, is recorded in C_1 format. The advantage to matching in the richer C_1 representation lies in the greater expressive power to represent complex stimuli, such as object specific parts or textures, while maintaining that low-level invariance to position and scale captured by the C_1 features. Experiments using a modified model where C_2 is computed by comparing S_1 representations without the intermediate C_1 stage show weaker discrimination between object classes.
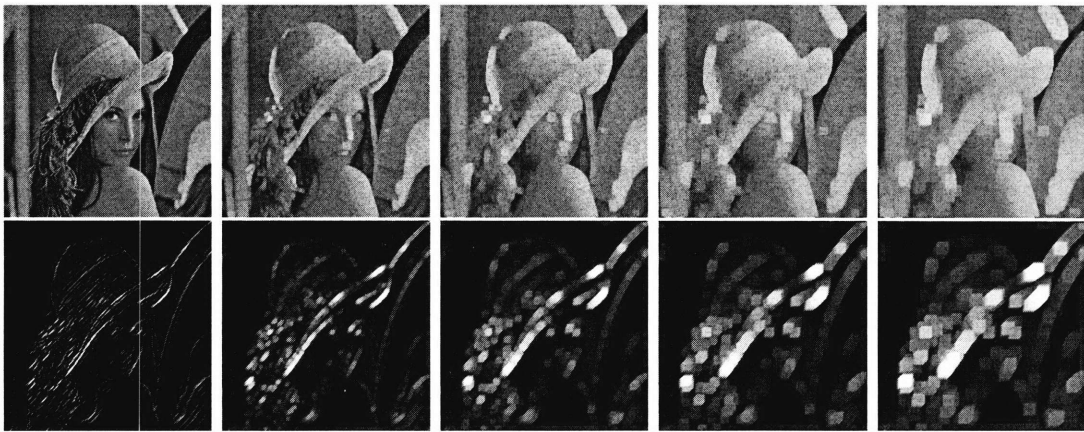
Figure 3-2: Max scale-space images of Lena (top row) and of the gabor filtered version of Lena (bottom row). While the gray-value image gets heavily distorted, the information in the sparse edge image is enhanced.
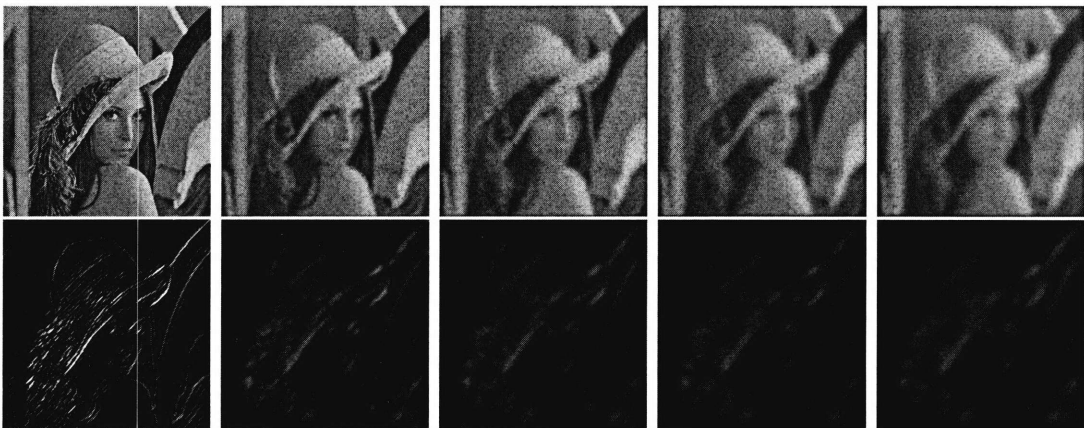
Figure 3-3: Gaussian scale-space images of Lena (top row) and of the gabor filtered version of Lena (bottom row). While the gray-value image degrades gracefully, revealing structures at different scales, the sparse edge image fades away.

Finally, the choice in the extent of the neighborhood size in computing C_2 is similar to discoveries in the patch-based representation literature regarding the importance of relative position information when calculating patch-based feature. Some objects, usually those with strong example-to-example correspondence, benefit from information about the relative position of the detected features. Limiting the pooling area for the maximum computation to that area of expected location results in improved performance. The cost of course is a loss of generalization to larger variations in pose and other common image transforms.

62

Object	car	pedestrian	bicycle	building	tree	road	sky	sidewalk	store
Type	shape-based			texture-based					

Table 3.2: The shape-based and texture-based object categories in the StreetScenes database

3.1.4 Computational Concerns

Currently, for one 960 × 1280 pixel StreetScenes image it takes approximately 300 seconds to compute the full set of StandardModel features in MATLAB on a 2GHz CPU with 4MB of RAM. It should be noted, however, that the implementation was not designed for maximum speed, and new implementations and approximations have improved this significantly. These efforts are current research in the lab, and publication will be forthcoming. Of course, the process is much faster in low resolution images

3.2 System Overview

It was decided at an early stage of this project to group the object categories in the StreetScenes database into the following categories: those objects which have spatial part-to-part correspondences, and those objects which do not. Loosely, given two examples of an object class and any point on one example, if a corresponding point can be accurately found on the other example, then this class exhibits such correspondence, and we will refer to the class as *shape-based*. If, on the other hand, the class does not have such correspondence, then we refer to the class as *texture-based*. One can certainly imagine object categories which push these boundaries, but for the most part the categories in our StreetScenes database can be devided unambiguously. Table 3.2 summarizes the StreetScenes objects.

In the system described here, the two types of object classes are handled using different learning strategies. Detailed descriptions of the algorithms for texture-based object detection and for shape-based object detection are given below, followed by appropriate performance measurements.

63

3.3   Shape-based Object Detection

In order to detect shape-based objects, the system presented here uses the C_1 features from the SMF set in combination with the well-known windowing technique. Windowing is used to enable the detector to recognize objects at all positions and scales, given that C_1 features have only limited position and scale invariance.

3.3.1   Classifier Construction

The training and testing data for these detectors is extracted using the cropping framework described in 2.3. Using only part of the StreetScenes database, leaving the remainder for future independent testing, square object and non-object crops were extracted for the three shape-based object classes and resized to 128×128 pixels. These crops were then converted into C_1 SMF space as detailed above in Sec. 3.1. In this way, each training example is converted into a $13,365$ dimensional vector. Both gentleBoost, a variant of boosting, and SVMs were used to model each binary problem. Results showing a comparison of the two methods for this problem are shown in table 3.3. These measurements were performed using the crop-wise paradigm outlined in Sec. 2.3 and the C_1 image features. Briefly, the crop database was randomly divided into $\frac{2}{3}$ training and $\frac{1}{3}$ testing, and SVM and gentleBoost classifiers were trained and tested for each split. The results show that the two learning methods perform statistically equally at this task.

3.3.2   Comparison to Baseline Algorithms

In order to get an idea of the relative performance ability of the C_1 based detectors, we compare them to classifiers trained on these same databases using other well known object detection techniques. We use the same crop-wise detection task as above, and report the results in Fig. 3-4. The thick dashed curve labeled "Grayscale" indicates the results of training a system using a simple grayscale feature vector instead of the C_1 values. In this system, each example is normalized in size and histogram equalized to build the feature vector.

64

	car		pedestrian		bicycle	
	Boost	SVM	Boost	SVM	Boost	SVM
AUC	99.2 ± 0.3	99.0 ± 0.3	90.4 ± 1.7	90.8 ± 1.8	98.5 ± 0.5	98.3 ± 0.6
EER	4.4 ± 0.8	4.8 ± 0.8	16.2 ± 2.1	15.6 ± 2.3	5.8 ± 1.6	6.4 ± 2.0
TP@FP=1.0%	88.6 ± 3.2	85.1 ± 4.6	41.1 ± 6.2	48.6 ± 8.7	77.7 ± 6.6	77.0 ± 6.2
TP@FP=0.1%	70.0 ± 7.5	60.7 ± 9.9	8.9 ± 5.0	6.0 ± 5.1	44.7 ± 20.9	46.3 ± 19.7

Table 3.3: Statistics comparing performance of gentleBoost with SVM for the C_1 based detection of cars, pedestrians, and bicycles on a crop-wise object detection task. Mean and standard deviation single dimensional measures are given for Area Under Curve, Equal Error Rate, True Positive Rate at False Positive Rate = 1.0%, and True Positive Rate at False Positive Rate = 0.1%. All statistics are displayed in percent score. We will show that the choice of SVM vs. Boosting leads to performance differences which are very small compared to the difference in image representation.

Another base-line detector is built using the Histogram of Gradients (HoG) features described by Triggs in [29]. This system is similar to C_1 in that it represents images using local oriented energy, but it also has major differences.

For a patch based system, we use a system similar to that described by Torralba in [117]. Each patch-based feature f_i is associated with a particular patch p_i, extracted randomly from the training set. The value of f_i is the maximum of the normalized cross correlation of p_i within a window w_i of the image. This window of support w_i is defined as a rectangle three times the size of p_i and centered in the image at the same relative location from which p_i was originally extracted. The advantage of these types of features over the gray-scale features is that the patch features can be highly object-specific while maintaining a degree of position invariance. For the results illustrated in Fig. 3-4, the system is implemented with $1,024$ such patch features with patches of size 12×12 in images of size 128×128.

Another system compared to in Fig. 3-4 is a part-based system as described in [67]. Briefly, object parts and a geometric model are learned via image patch clustering, and detection is performed by re-detecting these parts and allowing them to vote for objects-at-poses in a generalized Hough transform framework. While good results for cars were reported in the original work, we see that for the *pose-independent* learning problem, this patch-based constellation model is outperformed by the SMF system.

Finally, for a complete comparison, we include results for a detector of our shape-

based objects which is based on the C_2 features instead of the lower C_1 SMFs. This system calculates one C_2 value per patch for each image. We use a collection of 444 patches extracted at random locations from the training crops, 111 each from 4 different sizes, 4, 8, 12, and 16 pixels. We found that, for these objects, the detector benefitted from some notion of feature locality. Calculating the maximum S_2 response in a box around the relative extracted location of the prototype was in general better than using the maximum over the entire image. Figure 3-5 illustrates how the size of the maximization window, from S_2 to C_2, effects the classifier performance. It was also found that using crops specifically only from the positive training examples did not significantly improve performance over those crops which were extracted from both the negative and positive data.

For the SMF, "grayscale", Histograms of Gradients the "local patch correlation" classifiers, the statistical learning machine is the gentleBoost classifier. Performance is no better when using a linear or polynomial kernel SVM.

The classifiers employing the standard model feature C_1 based classifiers were dominant for the car and bicycle classes, and second best for the pedestrian, being beaten only by Triggs' Histogram of Gradients feature, which, it should be noted, was the object for which the Histogram of Gradients' parameters were tuned. The C_2 based classifier also performed strongly for the car and bicycle classes.

3.3.3 Full Shape-based Detection System

In order to build a system which is capable of detecting these objects at all positions and locations, the well known windowing technique is used, *i.e.*, a dense subset of all possible square windows is cropped from the test image, converted into the feature space, and passed through the classifier. The result is a real-valued detection strength as a function of location and scale. Local neighborhood suppression is then used remove redundant detections. The parameters and properties of the windowing and neighborhood suppression algorithm we use are listed in appendix B. In Fig. 3-6 we present some typical results of this type of detection. In order to quantify the fidelity of this sort of detector, we performed a box-wise detection task using 100 images

Car: Statistics						
	C_1	HoG	C_2	Part-Based	Grayscale	Patch Correlation
AUC	99.2 ± 0.3	97.6 ± 0.4	97.8 ± 0.4	93.0	96.0 ± 1.0	75.4
EER	4.4 ± 0.8	7.8 ± 1.2	7.7 ± 1.0	14.1	9.3 ± 1.5	31.3
TP@FP=1.0%	88.6 ± 3.2	66.3 ± 5.5	65.7 ± 5.0	20.2	73.0 ± 4.5	7.0
TP@FP=0.1%	70.0 ± 7.5	36.9 ± 9.8	33.8 ± 10.0	3.1	49.8 ± 8.0	1.3

Pedestrian: Statistics						
	C_1	HoG	C_2	Part-Based	Grayscale	Patch Correlation
AUC	90.4 ± 1.7	97.1 ± 0.8	94.4 ± 0.9	93.3	81.5 ± 2.2	85.9
EER	16.2 ± 2.1	8.1 ± 1.0	13.0 ± 1.7	14.8	24.9 ± 2.1	22.2
TP@FP=1.0%	41.1 ± 6.2	69.0 ± 6.2	47.5 ± 6.2	15.3	20.4 ± 4.0	14.7
TP@FP=0.1%	8.9 ± 5.0	39.7 ± 12.3	20.4 ± 6.5	4.7	5.7 ± 3.4	4.7

Bicycle: Statistics						
	C_1	HoG	C_2	Part-Based	Grayscale	Patch Correlation
AUC	98.5 ± 0.5	95.6 ± 2.1	96.0 ± 1.2	87.7	77.7 ± 3.6	70.2
EER	5.8 ± 1.6	10.2 ± 2.9	10.5 ± 2.8	19.1	28.9 ± 3.4	35.0
TP@FP=1.0%	77.7 ± 6.6	61.5 ± 7.5	51.4 ± 10.8	14.6	17.5 ± 7.8	5.5
TP@FP=0.1%	44.7 ± 20.9	27.7 ± 14.7	25.0 ± 10.2	5.4	4.8 ± 5.1	0.5

Figure 3-4: ROC curves and statistics comparing performance several different object detection systems using a crop-wise measure. The details of each system are described in the text. The table shows, for each system, the area under the ROC curve (AUC), equal error rate (EER), and the true-positive rate at two specific false positive rates. All statistics are shown in percentage. Standard deviations are not shown for the "Local Patch Correlation" or "Part-Based System" because these systems were tested only one time.
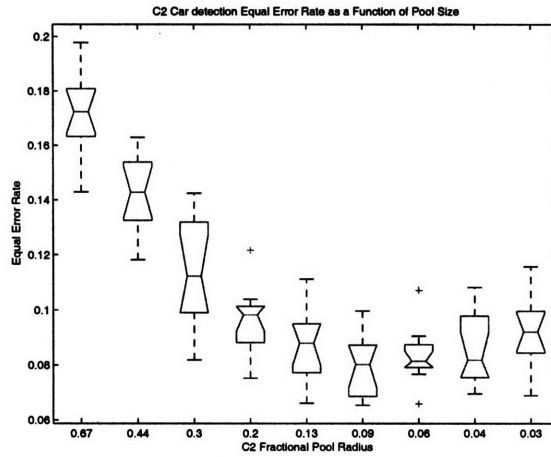
Figure 3-5: The performance of the shape based car detector using C_2 features depends on the size of the maximization window used to compute C_2 from the S_2 features. On the far right the maximization window is one pixel, meaning that C_2 is simply the maximization of the S_2 over scale, no spatial maximization takes place. On the far left the maximization takes place over nearly the entire spatial extent of the image.

distinct from those used in the above training and testing paradigms. The results of this experiment are illustrated via PR curves in Fig. 3-7. The grayscale detector is used as a baseline for comparison.

3.4 Texture-based Object Detection

Texture-based objects are those objects for which, unlike shape-based objects, there is no obvious visible inter-object part-wise correspondence. These objects are better described by their texture than the geometric structure of reliably detectable parts. For the StreetScenes database the detectable texture objects currently include buildings, roads, trees, and skies. It was decided that, at this stage of the project, the sidewalk and store classes were too dependant on contextual relationships to be detected simply by texture analysis.

68
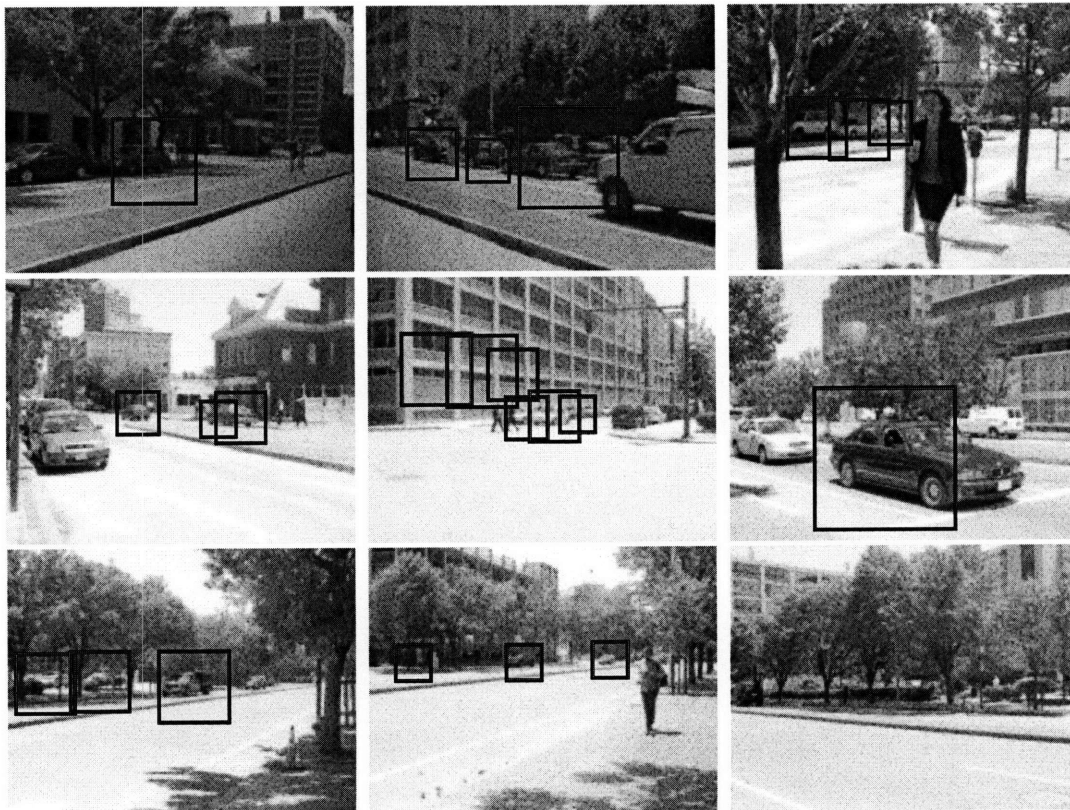
Figure 3-6: A detection task involving the StreetScenes shape-based objects. The C_1 features are used along with windowing and local neighborhood suppression in order to detect cars in full images. The system is charged with a false detection whenever the bounding box overlaps a unique ground-truth box by less than 50% of the union area. Several false-detections and false-negatives are visible in these images. Still, this task is very difficult, as there is no motion, and the cars may be at any pose without a pose-segmented training database. Note that the orange annotation square are difficult to see without a color display.
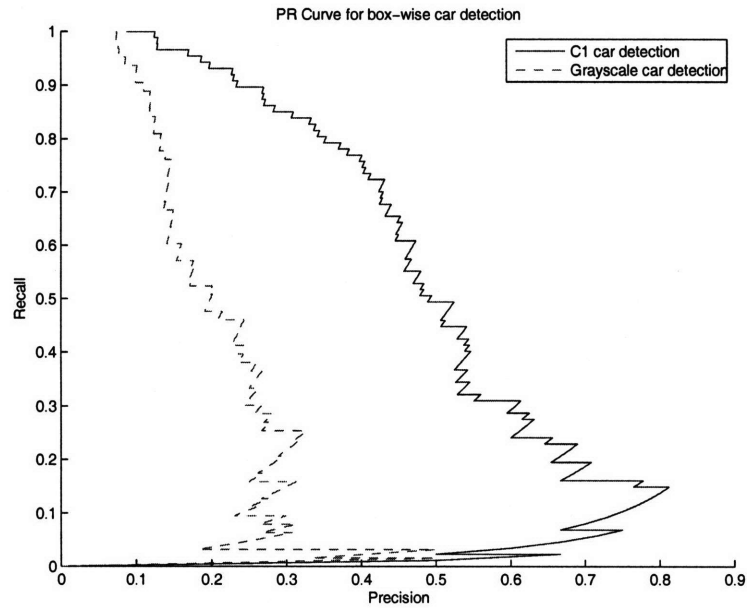
Figure 3-7: Precision-Recall curves for the car detection in natural scenes problem using the first 300 scenes as training and the next 100 as test. C_1 features are compared to grayscale features. From this curve we can surmise that with this implementation of the C_1 car detection system, in order to detect 50% of the cars we must suffer approximately 50% false detections.

3.4.1 Methodology

Texture based object detection systems were built in a pixel-wise manner, as described in Sec. 2.3. For the four object classes described above, locations of positive and negative object presence were selected from the training portion of the database. Locations of positive object presence for object type A were selected such that each such point was within a polygon labeled as class A, was no more than 15 pixels from the border of this polygon, and was greater than 15 pixels from the border of any polygon of a different class. This was done so as to prevent the collection of erroneous textures into the positive database due to object occlusion or loose labeling.

For each of these selected locations, a feature vector was sampled from the image. For instance, a simple version of the system might use the values of the three color channels as the feature vector for this pixel. We compare the relative performance of a number of different representations below, and illustrate their performance on real scenes.

Classifiers for each object were trained in a one-vs-all manner, where, for class A the positive examples from every class except A were used as negative training data. This choice was made because of purity concerns similar to those we had for the positive data. Boosting was used as the statistical learning machine of choice, but linear-kernel SVMs performed equally well empirically.

In our experiments we report two systems relying upon the standard model features. The first system uses a 32 dimensional vector of C_1 features (8 bands times 4 orientations). The second system uses a representation consisting of 444 C_2 features.

In order to better describe the process of collecting the training and testing data, let us describe the process of extracting one texture example in the C_1 representation. As described above, each sample is associated with a particular image and a particular location within that image. In order to build data for the C_1 system this image was first processed as in Sec. 3.1, resulting in a set of images, one for each orientation and scale-band. A vector is then extracted by interpolating a value from each such image at the same *relative* location, resulting in one value for each scale and orientation.

To collect C_2 values, we continue processing as described in Sec. 3.1, up to and including the S_2 layer. If we were then to take a maximum over the full S_2 pyramid for each patch, then each location in the image would have the exact same representation. We would like instead a representation in which local information is preserved. Thus, a variant of C_2 is used in which S_2 values are maximized only locally in scale-space. The version we use here uses a maximum response over all scale bands, but only a 15 pixel wide maximization window in the spatial dimensions. The effect is similar to having cropped out a small region of the texture object, and allowing responses only from completely within the crop. This is similar to the C_2 system for shape-based objects, as described in Sec. 3.3.2.

The C_2 implementation we used employed 111 features each from four possible patch sizes, resulting in a total of 444 features per pixel. The associated prototypes were extracted from random locations in the training image database. Since the boosting classifiers we use only reference one feature per weak learner, and, in general, boosting classifiers were terminated after 300 rounds of boosting, all classifiers did not use the full list of features.

3.4.2 Measures of Performance

The first 300 StreetScenes images were set aside for training and testing the system and benchmark systems. Again we used the pixel-wise object detection measure described in Sec. 2.3. These images were randomly assigned to either training or testing, and, as far as possible[2], 10 extraction locations were selected from each image for each object, resulting in nearly $3,000$ points for each object type. Using the methods described above, texture samples were extracted from the database, boosting classifiers were trained, and the results of testing these classifiers are displayed below in figure 3-8, along with the results of a number of baseline texture representations, described below. Subjective results showing the operation of these classifiers are displayed in figure 3-10, and an enhancement using segmentation is described in section 3.4.4.

---

[2]Some images do not contain pure examples of the target object type.

| | Building vs. Other | | | | Tree vs. Other | | | |
	AUC	EER	TP@FP=1.0%	TP@FP0.1%	EER	AUC	TP@FP1.0%	TP@FP0.1%
$SMFC_2$	93.6	12.4	44.0	13.2	n	n	n	n
$SMFC_1$	85.6	22.6	58.7	14.0	n	n	n	n
Blobworld	70.7	33.9	1.7	0.1	80.4	26.6n	2.9	0.1
Texton1	70.0	35.5	1.1	0.2	85.4	22.7	7.7	0.7
Texton2	73.8	30.2	8.3	3.5	87.9	19.6	11.7	2.0
EdgeHist.	74.8	31.5	3.8	1.4	79.3	24.5	6.3	0.1

| | Road vs. Other | | | | Sky vs. Other | | | |
	AUC	EER	TP@FP1.0%	TP@FP0.1%	EER	AUC	TP@FP1.0%	TP@FP0.1%
$SMFC_2$	96.8	7.9	79.1	30.0	98.9	3.8	91.5	54.0
$SMFC_1$	93.0	13.7	37.0	4.8	93.7	13.4	70.2	31.7
Blobworld	74.1	32.3	1.4	0.1	78.1	25.9	2.3	0.1
Texton1	86.5	21.3	18.8	4.9	92.1	15.9	53.5	1.7
Texton2	88.5	17.7	23.6	6.9	93.9	14.7	52.9	29.4
EdgeHist.	88.2	17.5	8.1	1.0	77.5	30.1	29.2	20.9

Figure 3-8: ROC curves measuring performance on a pixel-wise object detection task. Four binary texture classification problems are illustrated, using five different texture classification algorithms. The SMF based texture representations dominate the baseline algorithms.

In Fig. 3-8 we compare the results of the SMF texture-based object detectors against four other texture classification systems. The "Blobworld" system is constructed using the Blobworld features described in [24]. Briefly, the Blobworld feature is an eight dimensional vector at each pixel; 3 dimensions encode the color in the well-known Lab color space, 3 dimensions encode the texture using the local spectrum of gradient responses. The final two dimensions encode the (x, y) positions of the pixel. In order to be fair, the color and position information was removed from blobworld. Later we will display some results exploring the role of color and position information in the texture object detection problem. For this first experiment, however, we

concentrate our attention on texture alone.

The curves labeled "Texton 1" and "Texton 2" are are the results of a system based on [92]. The Texton feature is extracted by first processing the test image with a number of predefined filters and taking the absolute value. Texton 1 uses 36 oriented edge filters arranged in 5° increments from 0° to 180°. Texton 2 follows [92] exactly by using 36 gabor wavelet filters at 6 orientations and 3 scales. For both of these systems independently, a large number of random samples of the 36 dimensional edge response images are taken and subsequently clustered using k-means to find 100 cluster centroids. Each of these centroids is called a 'texton.' The 'Texton image' is calculated by finding the index of the nearest (in an L_2 sense) Texton for each pixel in the filter response images. The feature vector used for learning the texture-based object model is built by calculating the local 10×10 histogram of Texton values. The Texton feature is thus 100 dimensional, one dimension for each histogram bin.

Finally, the "Histogram of Edges" system is built by simply using the same type of histogram framework, but over the 36 dimensional directional edge response of "Texton 1" rather than the Texton identity. Therefore, each feature in the histogram of edges feature is a sum of the local filter response. This ca not be reduced into linear operation, however, due to the intervening absolute value. Learning for each of these features is done with 300 rounds of boosting over regression stumps.

From Fig. 3-8 we see that, while different methods have particular strengths for some objects, the SMF based texture system has superior performance on every object class. Changing the type of classifier from boosting to SVM does not change the order of the performances. Performing the test in a multi-class framework results in a correct-detection rate of 88%, where the baseline performance for random guessing would be 25%.

3.4.3 Color and Position for pixel-wise detection

With the regular structure of the StreetScenes, sky at the top, roads at the bottom, it is no surprise that position information can help in the classification task. Color information is also very helpful, as to be expected. To demonstrate, we have per-

formed a test identical to the one described above, but the feature vectors for each point were appended with either additional data representing the color in Lab space, or the (x, y) position of the pixel, or both. Figure 3-9 illustrates the results of these experiments. In order to avoid clutter, color and position are shown in combination with either the blobworld features or the C_2 features.

It can be seen that the relative utility of color and position information depends on the object type. Color is an especially important cue for detecting trees in these images, as can be imagined, since there are so few other objects which share their color space. For the road class, color is important, but not as important as the position feature. It should be noted that all classes benefitted by access to the C_2 texture features, more so than the other available texture features. The addition of color and position features improved the detection in multi-class score from 88% to 93%.

The fact that different features are important for different types of objects is a somewhat intuitively obvious, but yet important lesson. We will explore methods of intelligently combining feature modalities further in chapter 6.

3.4.4   Detection in Full Images

In order to detect the texture object categories in full StreetScene images, we simply apply the learned classifier to each location within the image. In the middle column of figure 3-10 we see the results of this process, where each pixel has been tinted the color corresponding to the object classifier that fired with the most confidence (brown for buildings, blue for sky, green for trees and gray for roads). As can be seen, the classifiers seem to be able to detect their preferred stimuli, but the results are rough and anomalous responses are common at the borders between two texture objects. One way to address these problems is to segment the input image (in the original space), and then smooth the classifier responses over each segment.

We used the publicly available image segmentation software "Edison" [26]. Within each contiguous segment, the results of each classifier were averaged. The assignment of pixels to object classes then proceeds as above, the results of which are shown in the right most column of figure 3-10. While the results are subjectively much more

Pixel-wise Building Detection / **Pixel-wise Tree Detection**

Building vs. Other | Tree vs. Other

	AUC	EER	TP@FP1.0%	TP@FP0.1%	AUC	EER	TP@FP1.0%	TP@FP0.1%
BW	70.7	33.9	24.5	1.8	80.4	2.7	2.9	0.1
BW+Lab	81.9	23.2	14.7	9.7	97.4	7.7	51.2	8.8
BW+Pos	79.4	26.8	5.5	1.5	87.0	20.0	7.8	2.1
BW+Lab+Pos	92.1	2.5	93.2	68.2	97.3	6.6	54.5	5.5
C_2+Lab+Pos	95.8	9.9	63.6	29.2	98.3	4.6	72.5	12.0

Pixel-wise Road Detection / **Pixel-wise Sky Detection**

Road vs. Other | Sky vs. Other

	AUC	EER	TP@FP1.0%	TP@FP0.1%	AUC	EER	TP@FP1.0%	TP@FP0.1%
BW	74.1	32.2	2.3	0.1	78.1	25.9	2.3	0.1
BW+Lab	92.6	15.0	37.1	1.8	98.9	4.8	72.6	30.0
BW+Pos	99.4	3.4	82.2	45.0	95.7	11.1	44.3	14.0
BW+Lab+Pos	99.7	2.5	93.2	68.2	98.8	3.4	91.5	53.9
C_2+Lab+Pos	99.8	1.9	96.9	81.5	99.6	1.9	96.2	81.0

Figure 3-9: ROC curves, similar to those in figure 3-8, measuring performance on a pixel-wise object detection task. Here we focus on the effect of color and position on the discriminibility of the texture classes. **Note that only the low false positive region is shown here, up to FP $= 1\%$.** The addition of color (*Lab*) and position (*Pos.*) features alongside the texture information provides the classifiers with better cues and improves performance, but not identically across object classes. *BW* indicates blobworld texture features, and C_2 indicates the standard model features described above. It seems that the *Pos.* features are important for roads and skies, but not for trees and buildings. Also, *Lab* features are helpful for skies and trees, but less so for buildings and roads. C_2 texture features are very helpful for buildings and skies, but not much better than *BW* features for the other classes, once *Lab.* and *Pos.* are taken into account.

appealing, the smoothing over segments was not found to reliably improve system performance, measured pixel-wise. One thing we would like to try is to weight the responses nearer to the center of the segment stronger than those responses closer to the edge, so as to remove the anomalous texture responses due to the border of the two objects.

3.5 Discussion

The system described in this chapter was designed to detect multiple object types in natural scenes. It was shown that the crop-wise discriminative classification of shape based objects is performed most accurately using the Standard Model C_1 features, while the detection of the texture-based objects, using a pixel-wise detection measure, was performed performed most accurately with the C_2 features. These two systems were compared with several baseline systems from the literature, including several which have shown state-of-the art performance in other databases.

We also have demonstrated that the standard model features, designed as a biological model, are very capable of strong performance in this scene understanding task. More than just support for the model, this suggests that computer vision should continue to take some inspiration from what is known of the biological system. At the same time, the types of processing done with the features, boosting and so on, is biologically implausible.

Whereas we use a discriminative approach to scene-understanding, through the detection of these objects, the system proposed by [111] detects objects, and their relationships to scenes, through a generative model. This model is capable of automatically learning objects, parts of these objects, and relationships between objects. The system is impressive in its structure and training methodology, but it appears, at least superficially and for the time being, that the discriminative system described here is more accurate. We look forward to more direct comparison in the future through the creation of public databases and measures like the StreetScenes database.

One more difference between this system and that described in [111] is that our
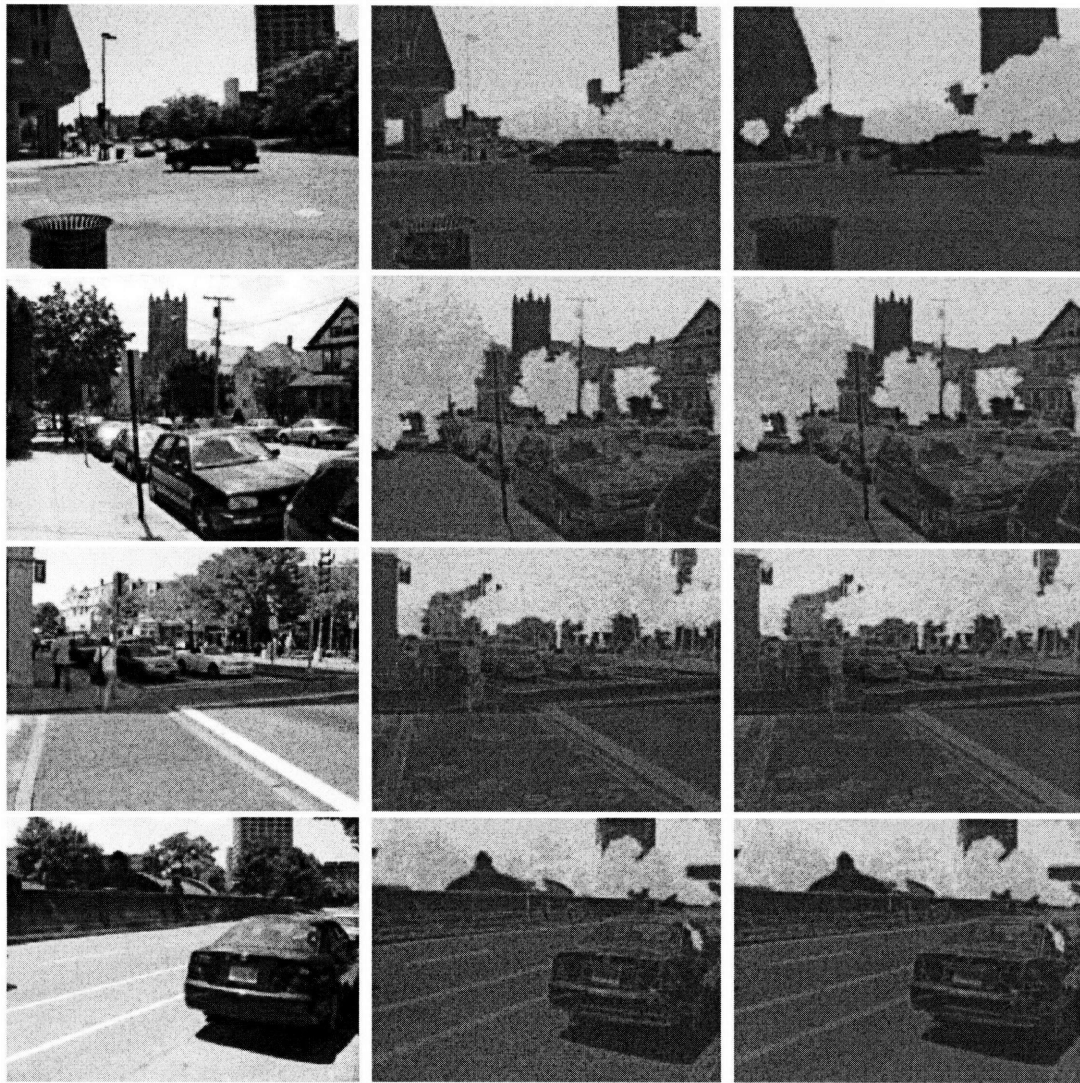
Figure 3-10: An illustration of the detection of four texture objects (buildings, trees, roads and skies) in four sample StreetScenes images. *Left*: Original Image. *Middle*: Texture object detection using local Color, Position, and the C_2 features. Color indicates the assigned label (brown, green, gray, and blue, respectively). *Right*: The same results after smoothing over segmentation. Again, these results are nearly impossible to interpret without a color display.

system does not make use of context, *i.e.*, it there is no way for objects influence the detection of other objects. In [111] objects may influence the detections of others probabilistically through the generative model. A contextual system for object detection in StreetScenes images is described in the next chapter.

Chapter 4

Contextual Modulation

In this chapter we will describe a simple feed-forward context feature and demonstrate its effectiveness across a variety of object types. By coupling a detector for each such object's appearance with a detector for that same object's *context*, we will support previous studies [117, 23] which show that, for at least the objects tested in our street-scenes database, context does aid in detection performance, especially when the appearance information is weak.

Using this model, we will explore some relevant issues regarding context. For instance, In Sec. 4.2.3 we address whether low-level context (derived from visual early features like wavelet values) is as powerful as high-level context (derived from semantic information like the presence of other objects). Previous systems have used one or the other, but this is the first direct comparison of the two.

The implications of this research question may be far reaching. If little or no benefit is gained by designing a system based on high level information, then context is nothing more than an additional classification feature, perhaps with a larger *receptive field*, and can be computed in a feed forward manner. If, instead, context is heavily dependant on high level information, then robust context-enabled object detection systems may be limited to computation structures involving some form of feedback.

We will also show that the utility of context information is related to the difficulty of the detection problem. If it is very difficult to discern the target object from the background, perhaps due to occlusion or low resolution images, then visual context

can be very helpful, but when the target object is unambiguously visible, then the context is only marginally useful, suggesting that the context information is highly redundant to the appearance information.

Note that, regarding these investigations, we make several assumptions. Firstly, in the experimental investigation, all tests use our StreetScenes database, and the utility of context is measured via improved detection of the shape-based objects in this database. It is likely that objects in other scenarios will have similar contextual relationships. Furthermore, all studies are performed using the same general framework for the contextual feature. The context feature captures semantic and/or low-level visual information sampled in a pre-determined spatial pattern. This feature is very similar in style to features which have been successful for describing object appearances for detection algorithms, and is an obvious extension thereof. All claims we make about context can only be supported in so far that this feature captures well the available context information. If there is pertinent, computable context information which this feature does not express to the subsequent learning machine, then the results of these experiments do not adequately answer our stated inquiries into the nature of context. It is hoped that this work will serve to tie together some of the disparate notions of context and serve as a resource to those who are interested in perhaps adding contextual level understanding to an object detection system.

4.1   A Feed-Forward Context Feature

Our system eschews complex models in lieu of a fast, simple, feed-forward classifier. We favor the discriminative approach to the learning problem, and as such are compelled to learn both relative and absolute geometric models in such a framework, as both are important to the context problem. For instance, we expect skies near the top of the image[1], but we expect cars to be above roads[2], no matter where the road appears. We show that there is an easy solution in using the right type of features.

---

[1]An example of the utility of *global* position information
[2]an example of the utility of *relative* position information

The construction of the context feature is best understood as a two stage process. In the first stage, the low level and semantic information is built. In the second stage, the context feature is constructed at each point by collecting samples of the previously computed features at pre-defined relative positions. This process is described in detail below.

4.1.1 Computing Low Level Image Features

To produce low-level "early" visual features, we first downsize the input image to 60×80 pixels and compute color and texture spaces as in Blobworld [24]. This resolution was selected as it was large enough to capture well the gross pattern of texture-objects (buildings, trees, roads, etc.), but small enough to work with computationally. Blobworld returns a new image represented with six layers of information, three for color, and three for texture. The color space used is the well documented CIE LAB space. LAB color is designed to model color dissimilarity as seen by humans as euclidian distance in the color space. The texture layers capture information about the local structure of the brightness gradient. The first texture feature is referred to as the polarity, and measures the likelihood of the local image gradient to switch direction. In a sense it discriminates between boundaries of brightness and distributed textures. The second feature is the anisotropy, which is roughly a measure of the relative strength of the gradient in orthogonal directions. Locations with large anisotropy have oriented energy in several directions, whereas locations with low anisotropy are limited to straight edges or parallel structures. The third texture feature is the texture contrast, which can be seen as a way to measure the roughness or harshness of the region, or how strongly the local brightness is changing.

In addition to the 6 color and texture features, we also include 10 features to represent the global position. These position features are calculated at each pixel $p_i = (x_i, y_i)$ by recording the L_2 distance from p_i to a set of 10 predefined locations $Z \equiv \{(x_j, y_j)|j = 1..10\}$. These locations Z are roughly evenly distributed over the image. This representation was chosen in order to make it possible for a classifier, even a simple linear one, to learn a wide variety of global position priors. For instance,

83

were just the x and y values recorded in the feature, it would be impossible for a linear classifier to prefer points near the center of the image over points near the borders.

4.1.2 Semantic Image Features

In order to investigate the importance of high-level features in constructing context cues, we include several semantic image features. For our task of detecting cars, pedestrians, and bicycles, we add four semantic layers indicating the presence of buildings, trees, roads, and skies. For instance, in the building feature, a pixel with a value of 1 indicates that this pixel is over a building, and a value of 0 indicates that it is not. Because of coarse labeling and ambiguous border cases, a pixel may be given multiple labels, i.e., it may be both building and tree, or it may have a null label. The ground truth for these four layers is available from the hand labeled StreetScenes images, as described in chapter 2 and illustrated in appendix A.

Since the ground truth semantic information is not available in *test* images, four binary support vector machine (SVM) classifiers were trained to automatically detect these categories. These classifiers were trained in the pixel-wise manner described in the previous chapter, but using the blobworld (including color) and position features described above, as opposed to the StandardModel features or other features described in Sec. 3.4. The training set for these classifiers consisted of $10,000$ samples per object category, and was extracted from 100 training images. Generating the semantic features for a novel test image involves first computing the low-level feature image, and then feeding this data into the four SVMs. See Fig. 4-1 for learned semantic labeling. Measures of the performance of these four classifiers are available in Fig. 4-3.

4.1.3 Building the context feature

At this point, the original image has been converted into an image with 20 layers of information. Each pixel p_i is labeled with 4 binary semantic features, 3 color features, 3 texture features, and 10 global position features. However, the context feature for
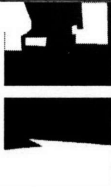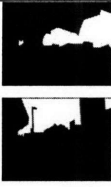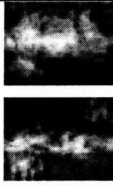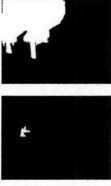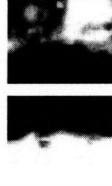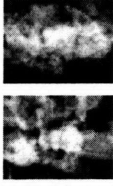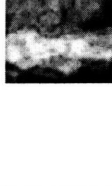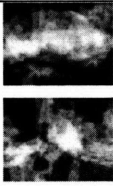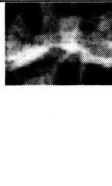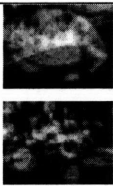
Source	True Semantic Label		Empirical Semantic Label		Learned Context	
Street Scene	Building Road	Tree Sky	Building Road	Tree Sky	Car Bicycle	Pedestrian

Figure 4-1: Column 1: Test images from the StreetScenes database. Column 2: True hand-labeled semantic data for the building, tree, road, and sky class. Column 3: Automatically classified semantic data. (Locations with larger positive distance from hyperplane shown brighter). Column 4: Learned context images for the three object classes: car, pedestrian, and bicycle. Brighter regions indicate that the context more strongly suggests objects presence.

p_i must hold information representing not only the immediate proximity, but also the larger neighborhood. This information is captured by sampling the image at 40 predetermined *relative* locations, as shown in Fig. 4-2. The relative positions are arranged in a quasi-log-polar fashion so as to sample the local neighborhood more densely than distant regions. This is similar to biological systems, such as the mammalian retina, and several computer vision systems, e.g., [6]. Specifically, data is sampled at 5 radii of 3, 5, 10, 15, and 20 pixels, and at 8 orientations. These 40 samples are concatenated to generate an 800 dimensional context feature vector for each pixel. Note that the 20 dimensional image is smoothed first by averaging over a
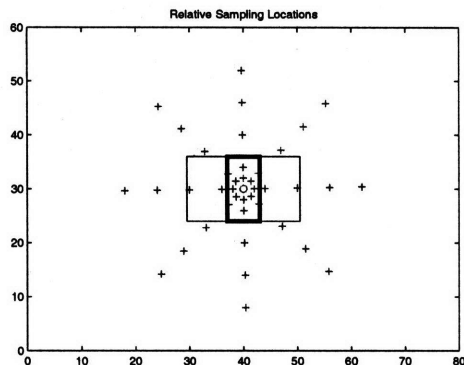
Figure 4-2: An illustration of the 40 relative pooling locations, plotted as blue '+' signs, relative to the red o . The thin black rectangle represents the average size of the cars in the database, and the thick black rectangle represents the average size of pedestrians.

5×5 window.

While it may seem that computing semantic-level information from the low-level information, and then including both is an exercise in redundancy, we should point out that this is not the case. Reductio ad absurdum, this argument would support the claim that one should only include the original pixel-level information, since all visual features can be computed directly from these. Since the automatically learning of appropriate data representations is still an unsolved problem, it makes sense to include all useful representations of the input. Greater attention to the problem of how to make use of multimodal data will be given in chapter 6.

4.2   Experiments and Results

We will compare the detection ability of our context detector to that of an appearance based detector. Afterwards, we will show that the two can work together to achieve levels of performance that neither classifier is capable of alone.

4.2.1   Fidelity of Semantic Information

Four SVMs were trained to discriminate between positive and negative examples of the four classes: building, tree, road, and sky in order to provide semantic information

86
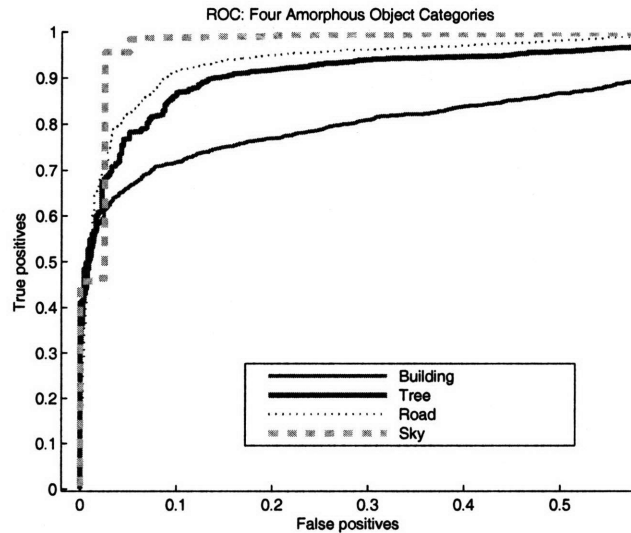
Figure 4-3: ROC curves for the four semantic classifiers; building, tree, road, and sky. These curves are a bit lower than those in Ch. 3 because they are not using the C_2 texture features. The advantage is that the computation is faster. It will be shown that these classifiers are good enough to be of semantic use to the shape based object detectors.

to the classifiers. The features used to learn these classes are the color, texture, and global position information described in Sec. 4.1.1. By splitting the training data and using cross validation, we obtain the ROC curves illustrated in Fig. 4-3. A similar one versus all approach to that used in Sec. 3.4 is used. While the learned semantic classifiers are not perfect, they are operating at a level much better than chance.

4.2.2   Performance of the context detector

Once the 100 images selected for the training of the semantic classifiers are removed, $3,447$ labeled images remain in the StreetScenes database. This corpus is split evenly into context-training and context-testing sets. To learn a model of object context, it is necessary to collect a database of samples of positive and negative context. One sample of positive context is taken per labeled target object in the training database. For instance, for each labeled car example in the training database, one 800 dimensional sample of positive context is extracted at the approximate center of the car. Additionally, ten times as many locations of *negative* context are recorded from locations at least 7 pixels away from our target object (in the downsized 60×80

pixel image).

Models of context are built by training a boosting classifier, and the performance is evaluated pixel-wise on the testing data. The results are plotted in Fig. 4-4. For comparison, we include results for a similar context detection system where the SVM-estimated semantic features have been replaced with true hand-labeled semantic features (in both training and testing situations). We also include results for a detector of object *appearance* trained from the same object examples. A description of the structure of this appearance detector is available at the end of this section.

By comparing the ROC curves, it can be seen that the advantage of having true semantic information, as opposed to the empirical semantic information output from the classifiers, is negligible. The appearance detector in general outperforms the context detector, although in the very high detection rate region the context-detectors are performing slightly better. This might be of use in a classifier cascade situation, where it is important to not remove true positives. To surmise, however, from these plots that the appearance detector is *better* than the context detector is wrong for the following reason: the measure used here is a measure for object detection, not object context detection. If, for instance, the context detector responds strongly to pedestrian context over a crosswalk, a location likely to have pedestrians, and there are in fact no pedestrians in the test image, then by this measure the context detector has performed poorly, i.e., a false positive. The proper way to measure the context performance is to quantify how much aid is given to subsequent stages in the detection process. This is studied in Sec. 4.2.4.

The Appearance Detector The appearance detectors for the three object classes are all constructed via linear kernel SVMs. Training examples for these classifiers were selected from the StreetScenes data using the same methodology as for the context detector, as described in Sec. 4.2.2. In brief, positive and negative samples of object appearance are extracted from the training images by selecting appropriate locations and sizes. For each training example, the minimum square bounding box of the object is calculated, and widened by a factor of $\frac{1}{6}$ on all sides. This bounding box is cropped from the image, converted to gray-scale, and resized to exactly 64×64

pixels. The resulting image is linearly filtered with 6 filters: four 3×3 Sobel filters at $45°$ intervals, one 3×3 Laplacian filter, and one identity filter. After taking the absolute value of the result, each result except the identity filter is submitted to the morphological gray-scale dilation operation using the 8-neighbor model of connectivity and a radius of 5 pixels. Finally, the images are downsampled to 16×16 using bilinear filtering. The resulting $1,536$ dimensional data ($6 \times 16 \times 16$) is used to train the SVM. In a windowing framework it is possible to filter and dilate the whole image before the actual windowing step, so as to save computation time. This quick, crude approximation to the Standard Model C_1 feature has been shown to achieve nearly equal levels of performance while requiring much less computation.

4.2.3   Relative Importance of Context Features

Previous systems employing context to aid in the object detection task have used either low-level image statistics or high-level object detections as their contextual clues. In this experiment we wish to answer whether the features input to our context classifier need to be of high level, or if, instead, the low level information is sufficient.

We train four context classifiers using the system outlined above in Sec. 4.1, the only difference between the four being the subset of the context features used. Classifier 'A' uses only global position information and nothing else, classifier 'B' uses only semantic information and nothing else, classifier 'C' uses only color and texture information and nothing else, and finally classifier 'D' uses both color-texture features and semantic features, but no position information. The ROC curves of these classifiers are illustrated in Fig. 4-5 using the same measure as the previous experiment. We see from the figure that the position based classifier performs much better than chance, even though position information is identical in every test image. This performance is to be expected since most cars and pedestrians are near the bottom half of the image; the distribution is not uniform. The position-only-classifier can be considered to be calculating a sort of discriminative object prior for position. We see also that the semantic-information based classifier performs at about the same level as the position detector, even though this detector is not privy to the position information. We are

89
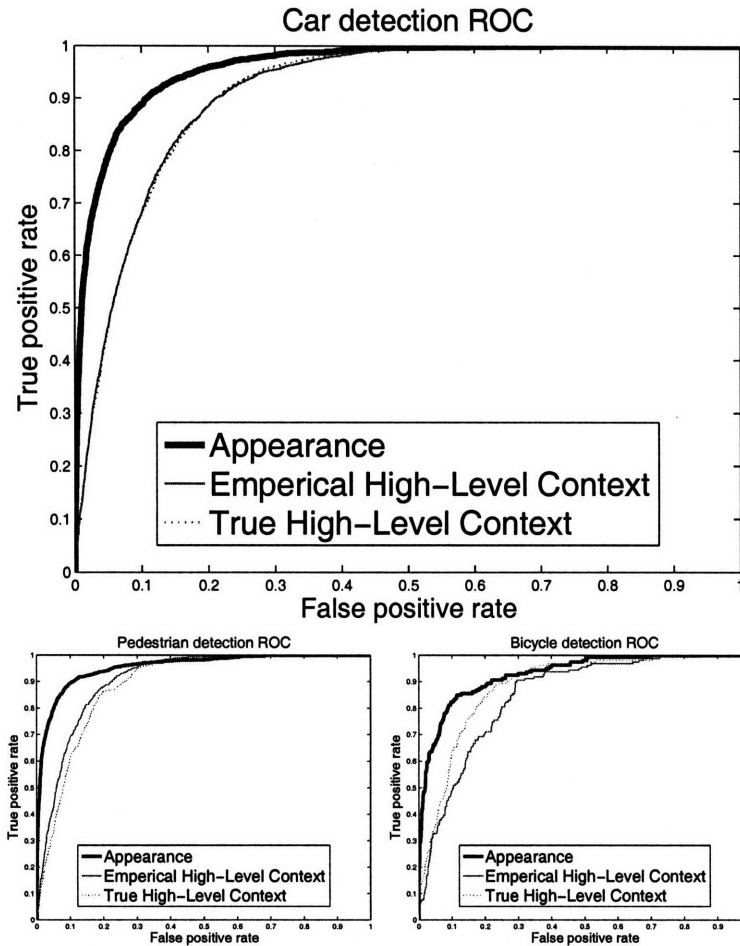
Figure 4-4: ROC curves for car, pedestrian and bicycle detection using (solid red): context with estimated (learned) semantic features, (dotted red): context with hand-labeled semantic features, and (solid blue): appearance. In the car example the two context curves overlay each other, and are hard to discriminate. From these measures we see that the appearance is a more reliable signal of object presence than the context.

surprised, however, to see that the low-level-feature based detector performs better than either of these. It was presumed that information about the *relative* positions of large objects within the image would be the best cue as to the likely location of the target objects. This result would be of definite interest to practitioners, who may invest a great deal of time into complicated contextual classifiers and may be disappointed by the benefit over simpler methods.

Also of note is that the classifier which uses both semantic features and color-texture features does only marginally better than the classifier which uses only color-
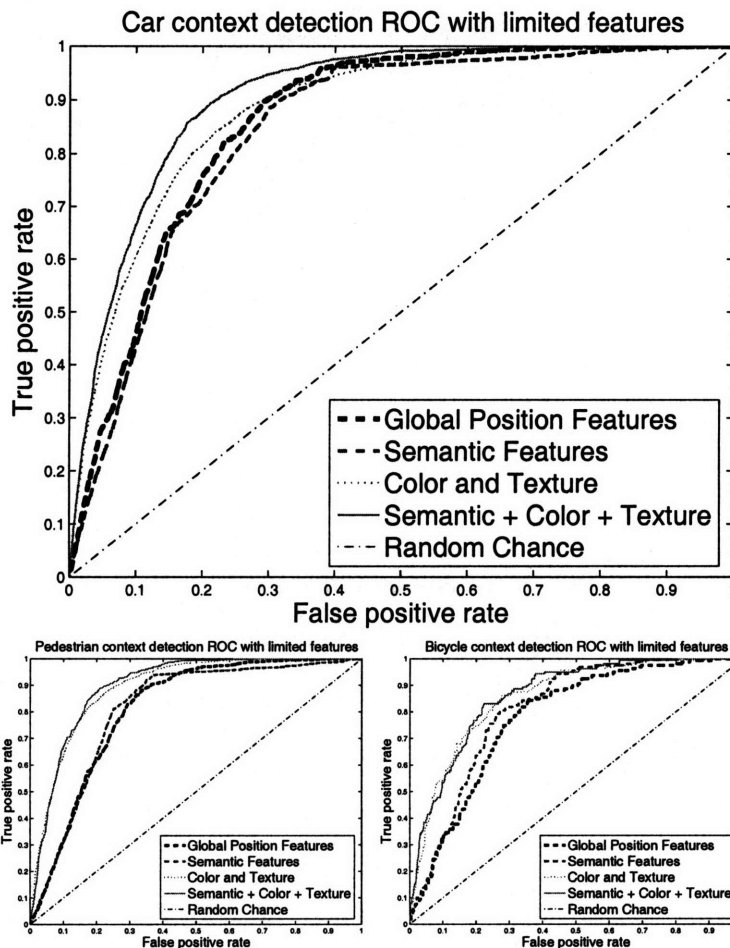
Figure 4-5: Different modes of contextual features have different utility to the detector. In all cases the low-level color and texture were the strongest cues. The performance of the system when using only semantic features is approximately equal to the performance when using only global position within the image.

texture features. This suggests that almost all the relevant information available from these semantic features is also immediately available from the color-texture features. Results are not improved by using the true semantic information in place of the empirical semantic information.

One might notice that the system samples contextual information from some locations which may overlap the target object. It is possible that what is being learned is less a model of context and something more akin to a model of appearance. This would explain why the low level image description appears to be more influential than the high level information. In order to test this hypothesis, an experiment is performed
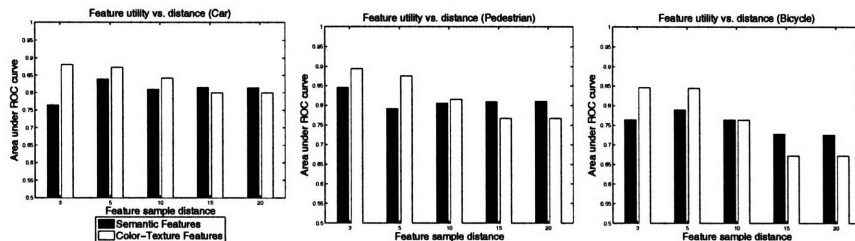
Figure 4-6: The quality of a detector of object context depends not only on the mode of features used, but also on the distance from which these features were sampled. For each object, context classifiers were trained and tested using either high or low-level features at $d \in \{3, 5, 10, 15, 20\}$. Each bar in this graph illustrates an area under an ROC curve of such a detector. As features are sampled from further away, the high-level information becomes more important than the low-level information.

illuminating the relationship between the importance of a feature mode and its distance d from the point of interest. In this experiment, for each $d \in \{3, 5, 10, 15, 20\}$, a classifier is trained with only low-level or high-level features from exactly distance d. No global position is included. Recall that the full image resolution at this stage is only 60×80, so these distances represent a wide receptive field. The results of this experiment are illustrated in Fig. 4-6. Plotting the area under the ROC curve for these classifiers illustrates that for all three objects tested, as one takes relative locations further and further from the target object, the semantic features become more important than the color and texture features. However, the low-level features retain much of their discriminative power even at great distances from the target object. Paraphrased, knowing the color and texture information at a few points very distant from a point of interest is about equally useful as knowing whether those same distant points have skies, buildings, trees or road, at least for the task of deciding whether the point of interest is likely to have a car, pedestrian, or bicycle.

The impact of these studies is pertinent to anyone implementing a contextual system to aid in the detection of objects. If early visual features can be used in place of high-level semantic information, then tremendous time can be saved by not labeling data and training classifiers to detect the neighboring objects. Instead, all the relevant information is already available with simple image transformations and a larger receptive field.
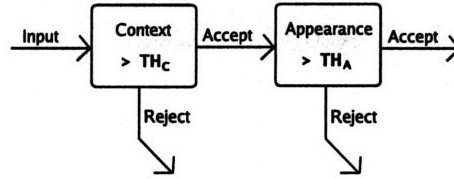
Figure 4-7: A data flow diagram of a rejection cascade combining both context and appearance information. Inputs are classified as positive only if they pass both classifiers. By tuning the confidence thresholds TH_C and TH_A, different points are achieved in the ROC plane.
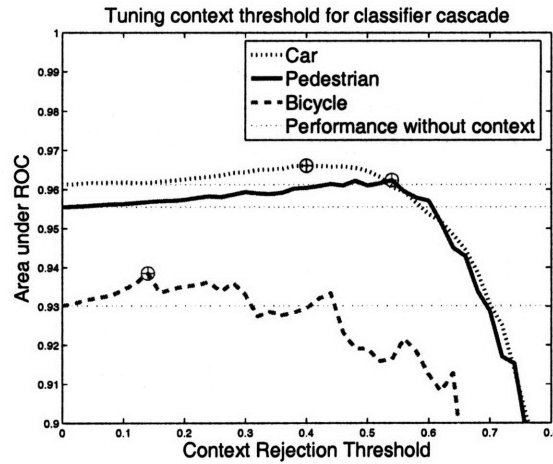
Figure 4-8: Area under the ROC curve of the rejection-cascade as a function of TH_C. The dotted lines appear where the curves intersect the line $x = 0$, signifying the performance level without any contextual assistance ($TH_C = -\infty$).

4.2.4   Improving object detection with context

In this final experiment it is demonstrated that context can be used to improve system performance. The architecture we will use is the rejection cascade illustrated in Fig. 4-7. In order to detect objects in a test image, the context based detector is applied first. All pixels classified as object-context with confidence greater than the confidence threshold TH_C are then passed to the appearance detector for a secondary classification. A pixel is judged to be an object detection only if the pixel passes both detectors. The context confidence threshold which maximizes the area under the ROC curve of the complete system is selected empirically using a validation set of 200 images. Fig. 4-8 illustrates the effect that the TH_C has on the performance of the detector cascade for three different objects. On the far left, $TH_C = 0$ corresponds

93
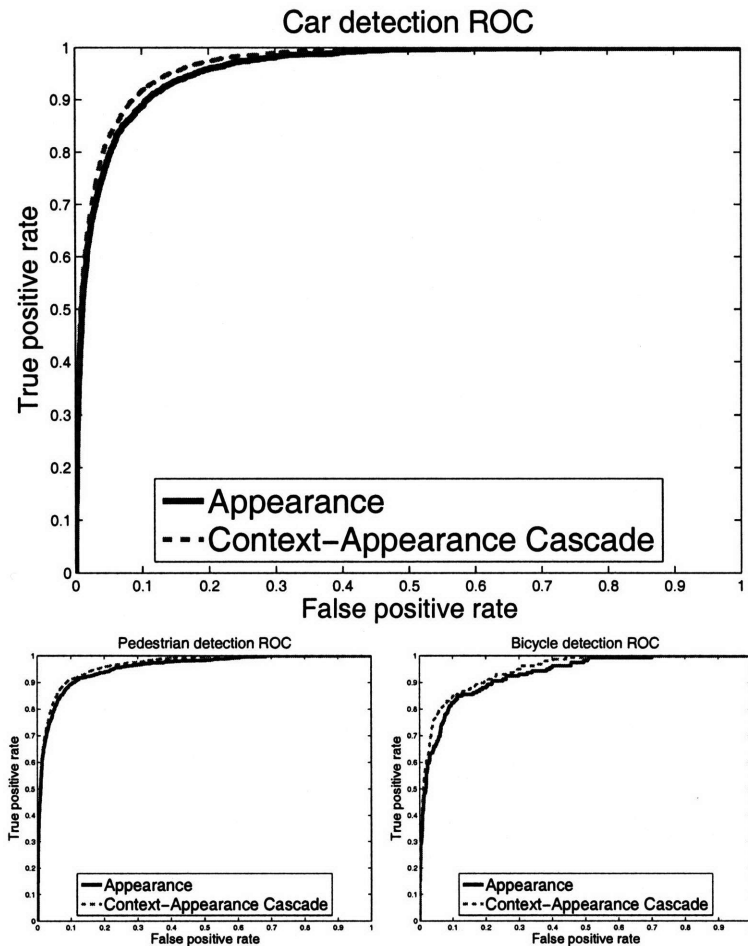
Figure 4-9: The car rejection cascade incorporating both context and appearance information outperforms the system using appearance alone. The level of improvement is, however, minimal. In this case it is hard to justify the additional complexity of the additional contextual analyzing unit by the additional performance levels of these detectors alone.

to not using context at all. At some point, as the threshold gets higher, the context classifier is rejecting too much and the performance drops due to false negatives. Somewhere between 0 and this level is where the context classifier can offer maximum assistance for this task.

ROCs of full system performance are illustrated in Fig. 4-9. These curves suggest that using context as a preliminary filter for an appearance detector may be a valid strategy, but, at least in this case the performance gain is marginal. The reason why the context cue was of so little assistance in this experiment can be understood by inspecting the distribution of the data. In Fig. 4-10 we plot the car examples in the

94
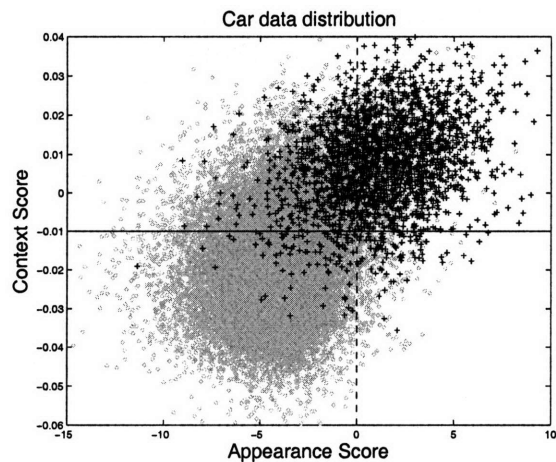
Figure 4-10: Empirical distribution of the car data in the appearance-context plane. Positive points are black '+' signs, negative points are the pink 'o' signs. Selected thresholds for the context and appearance detectors are shown as solid and dashed lines, respectively. Note that few points are simultaneously strong in appearance but weak in context.

plane where the x axis is the empirical appearance score, and the y axis is the empirical context score. A system trained to discriminate based on appearance alone would classify examples by setting some threshold along the appearance axis. In Fig. 4-10 the appearance classification boundary is illustrated with the vertical dashed line. The boundary for the context classifier is illustrated similarly as the solid horizontal line. Points which are classified positively by both systems lie in the upper-right quadrant of the diagram. The context classifier aids the system by rejecting negatives which are strong in appearance but weak in context, e.g., the negative points in the lower right quadrant. The important point to notice is that the distribution is curved such that these points are very rare. There are few samples, positive or negative, which *appear like the target object* and are simultaneously *out of context*. It is for this reason that the context cue did not give much performance gain. A superior appearance-based detector would achieve better horizontal separation of the positive and negative points, further marginalizing the importance of context. We attempted to use several other models of classifier combination, including training a linear model, but results were similar. Note that if boosting is used on the appearance and context together, concatenated into one feature vector, then the classification performance is even worse than just using appearance information alone, suggesting that in this

scenario the appearance information is much more relevant to the detection problem.

Further support of this thesis can be seen from the results published in [117], where for the three target objects *computer mouse, keyboard,* and *monitor,* the context is marginally helpful for the detection of the monitor, somewhat helpful for the keyboard, and very helpful for the detection of the mouse. Since the mouse is physically small it is difficult to detect without the contextual cues, but the monitor is visually unambiguous. For our application, there is very little appearance ambiguity similar to the monitor case.

4.3   Summary and Conclusions

The context system described in this work is simple enough for others to use in their own work and general enough to function across several object types. Experimental results demonstrate effective context detection for cars, pedestrians, and bicycles, and furthermore show that these context detections can be used in a rejection cascade architecture to improve detection accuracy. Our system's feed-forward design makes it possible to determine a map of object context at a resolution of 60×80 in under 10 seconds using a standard desktop computer.

It is commonly assumed that contextual cues can do much to improve the accuracy of an object detection system by eliminating false positives that fall out of context. We have demonstrated that visual phenomena which bear strong visual resemblance to the target object while simultaneously being out of context may be very rare, and in this case the benefit to be gained by such a combination is marginal. Instead, we propose that context is a useful cue for robust object detection *only* when the appearance information is weak, such as in critically low-resolution or very noisy images. In retrospect, it should not be surprising that context is useful in some situations and not useful in others. In this light, support for our context classifier could be made greater by building an experiment wherein the appearance data for the target objects have been in some way corrupted at the image level, *e.g.,* through blurring or some local nonlinear filter.

Our investigation into the relative importance of different modalities of context features is the first of its kind. Common wisdom suggests that context must be computed at a high level by inferring likely target object locations from the locations of other related objects in the scene, but our experiments show that accurate context can be determined from the low-level early visual features both near and far from the location of interest. It is hoped that other practitioners will take note and attempt simple contextual methods before building detectors for related objects.

Chapter 5

Extending StreetScenes: Novel Gestalt Image Features

5.1 Introduction

Computational approaches to perceptual tasks, visual and otherwise, can be divided into two major components: representation of the input signal, and learning, e.g., optimization, modeling, or estimation. Generally, the current systems exhibiting the highest levels of performance use standard learning techniques. The key to their effectiveness is either a very large training set, or an effective signal representation.

For visual problems, a wealth of image representations have been proposed in the literature. But, as mentioned in the literature review in Ch. 1, there has been rapid convergence into a few selected approaches. The representation of the image as a collection of gray value patch similarities or as a collection of edge orientation statistics are perhaps the most common. It is clear that these representations do not explicitly capture more intricate concepts, such as line continuity across long regions, which may be critical to the subsequent learning stages. Furthermore, the types of standard learning techniques commonly used for this sort of object recognition problem are unable to automatically generate these higher level information sources.

Our goal is to implement mid-level image representations based on *Gestalt principles* [3]. If these new representations make explicit visual signatures which discrim-

inate between examples of object and non-object, then we can expect system-wide performance gains. This performance constraint is critical; the literature has many examples of symmetry detectors, saliency detectors and other mid-level features [55, 1], however, few have been shown to add to the discriminative power of a state-of-the-art detection system. The task of creating Gestalt-based features that improve upon cutting-edge descriptors remains mostly unexplored.

Computationally, the implementation of these new features resembles the generalized Hough transform, but there are many significant differences, such as replacing summations with morphological operations. The importance of these non-linearities will be demonstrated experimentally. Also, in the classical generalized Hough transform, thresholding is used to detect salient image structures. Instead, we pass a reduced version of the voting space directly to the classifier. This design methodology will be elaborated as the paper progresses.

In Sections 5.2 through 5.5, we describe and evaluate new image descriptors based on the continuity, convex form, and parallelism Gestalt grouping principles, respectively. Caltech 101-objects database performance is demonstrated in Sec. 5.6. We then delve further into the properties of these types of features with explorations in parameter tuning and randomly generated features.

5.2   Continuity based image descriptors

The detection of continuous edges has long been a focus of the vision community, having been a topic of discussion in studies of saliency, segmentation, and boundary detection, etc., e.g., [104, 58]. Even the venerable Canny edge detection system [22] presupposes a notion of continuity to justify the use of two different thresholds and a hysteresis based detection scheme. While many modern object detection systems use edge gradients as feature vector elements, few have explicit encoding of the presence or absence of long continuous edges.   We will show that classifier performance can be improved over and above the levels achievable using state of the art histogram-of-edges features.

100

Given an input image I, apply the following procedure:	
*F*or each orientation θ, repeat steps 1-4 below:	
1. Image rotation: Rotate the image by an angle of θ.	$\circlearrowleft_\theta$
2. Initial filter: Take the absolute derivative of I in the x direction.	$\lvert I_x(x,y)\rvert$
3. Perform the following sequence of morphological operators on I_x:	
a. Local minimization with a kernel of length 7 in the x direction.	$\ominus(7,1)$
b. Local maximization with a kernel of size $(x,y)=7\times3$.	$\oplus(7,3)$
c. Perform a 2×2 subsampling.	$\downarrow(2,2)$
d. Record the current continuity data.	$R(x,y,\theta,\lambda)$
f. Increment λ.	
f. Return to 3a unless the data is less than 7 pixels wide.	
4. Bilinearly resize R to an appropriate size.	$\Downarrow R$

Figure 5-1: Pseudocode for the Continuity descriptor. In the real implementation the filters and not the image are rotated. The resulting description is parameterized by image locations x, y as well as orientation θ and line length λ.

The traditional approach to detecting continuous lines in an image is to use the Hough transform. First, gradient magnitudes are computed and thresholded at each pixel location, creating a set of edge-candidates. Each candidate is then mapped into the space of parameters of all lines which pass through the image. Traditionally the two dimensional parameter space (r, θ) is used, where θ is the slope of the line, and r is the distance from the line to some predetermined point in the image. Since each edge element could potentially stem from many different lines, the elements are allowed to cast many votes into the line-parameter space.

It would certainly be possible to then take the descretized parameter voting space, and use it directly as an image feature. This is unsatisfactory for several reasons, including the inability to represent lines of different lengths and the diffuse nature of the information about the actual spatial location of the lines within the image plane. For instance, a horizontal line in the top left may have the exact same representation as a horizontal line in the bottom right. The feature we propose below aims to address some of these issues.

5.2.1 The min-max continuity descriptor

In Fig. 5-1, we describe the algorithm used to calculate the image continuity features. The computation consists of first filtering the image with oriented filters, rectifying
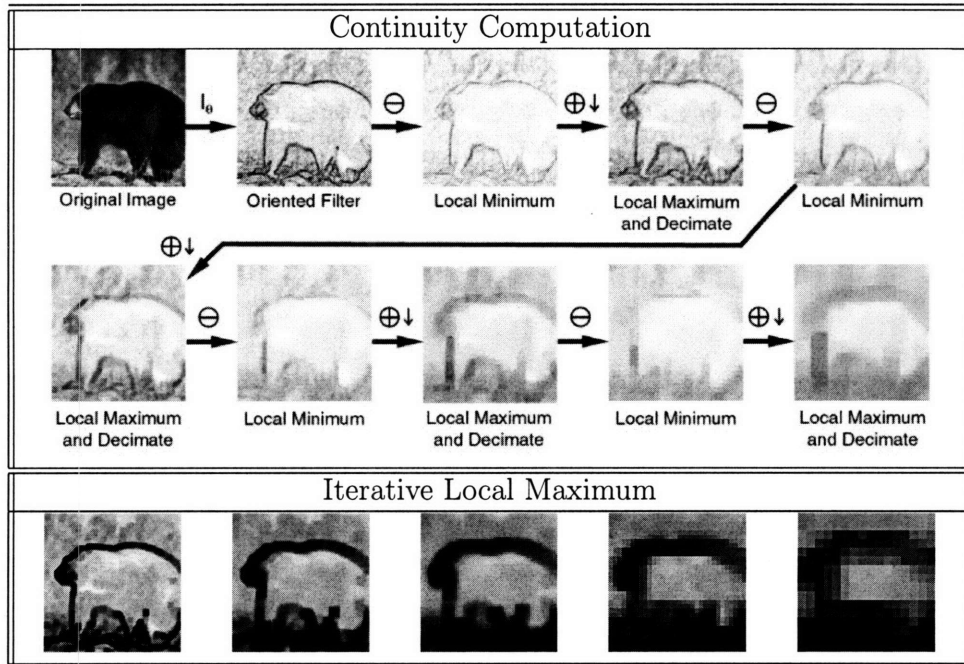
Figure 5-2: *(This figure is meant to be viewed in color.)*: *Top:* An illustration of the continuity computation. After the original image, different colors indicate different orientations of contours, red is vertical, etc. Deeper color saturation indicates stronger contour evidence. Note that as the computation proceeds, the texture is filtered away and only the correct tint of the long contour remains. *Bottom:* In contrast to the continuity detector, it is possible to simply iteratively take local maximums in the orientation space. The result, as shown here, is that all strong signals are spread out, no matter their local support. Since filters at many orientations are stimulated by strong edges, the result is that all orientations are strongly stimulated near the border of the bear.

the output, and then processing the result with rounds of local minimization, maximization, and decimation steps. Each round defines a discrete step in a scale space of continuous edges, wherein representations of longer contours are constructed as concatenations of smaller co-linear contours. The output is a feature vector in which each element describes the maximum magnitude of all ridges of a certain minimum length and orientation within a particular location in the image. After each iteration of step 3, this minimum length increases and the location specificity decreases. Intuitively, the minimum filter only gives a strong response if all edge elements in a contour are strong, and the maximum filter propagates that signal to the neighborhood. The wider maximum support is used to help detect straight edges which are not precisely at the represented orientations, and edges which slowly curve. When

the maximum support is longer than the minimum support, the feature becomes tolerant to small gaps in the contour at the cost of detecting more spurious contours in textured regions.

The current MATLAB implementation of this feature costs approximately 3 seconds per 128×128 pixel image, computing continuity in 18 orientations and at 6 scales. Since each orientation is computed independently, the cost is linear in the number of orientations. The erosion and dilation steps are computed efficiently using a data rearrangement and maximum procedure, but still the computation cost depends heavily on the size of the kernel used. The cost is linear in the area of the image. This computational cost assessment, and all others in this chapter, are calculated using a 2 GHz processor with 4 GB of memory running MATLAB version 7.1.0.183 (R14) in Unix.

For illustration, Fig. 5-2 depicts the continuity feature applied to an image of a bear: different colors indicate different contour orientations. Note that long continuous contours become enhanced and remain represented through gross image downsampling, while textured regions do not. The images in the bottom row indicated the results of a simple iterated maximization, without the erosion step. It is easy to see that the continuity detection process retains information about contour orientation, maintaining strong levels only when the support indicates a long continuous edge, while the local max process ends up simply spreading the strong gradient information in a local neighborhood, independent of whether the peak gradient measurement was part of a long contour, or rather a spike of noise in a textured region.

In morphology, image maximization and minimization are termed grayscale image dilation and erosion [109]. The continuous contour iterative step is similar to the morphological *opening* operation, except that the structuring elements are slightly different for the erosion and the dilation phases, and that a decimation takes place at the end of each iteration. Morphological operations have been studied for a long time in computer vision, and their properties and efficient computation methods are well known. Although the continuity operator is different from opening, we can use morphological image analysis to help understand some properties of the algorithm.

103

Property 1 (Noise Removal) *In each iteration the continuity operator will remove edge responses which do not maintain their strength for at least k successive pixels, where k is the diameter of the minimizing structuring element.*

Property 2 (Gap Tolerance) *Consider a long contour with a gap of g pixels. If the length of the contour is at least k on both sides of the gap, where k is the diameter of the erosion structuring element, then the gap will not be larger after the min-max iteration. If the contour is long enough, then the influence of the gap will diminish in successive iterations due to the decimation phase.*

5.2.2   Experimental validation

The utility of the continuity feature in object detection tasks is demonstrated in three separate object detection experiments. In the first two, the performance of an existing system is improved by simply adding the continuity features to the feature pool available during the learning stage. The third, experiments on the Caltech 101 object database, is described in Sec. 5.6. The first experiment demonstrates improved results in detecting cars, pedestrians, and bicycles in the StreetScenes database as described in Ch. 3. The second experiment involves discriminating images which contain an animal from those which do not, where the animal may be at any scale or position within the image, and windowing is not used. Finally, improved results are demonstrated on a three-class database consisting of low-resolution images of cars, mid-size vehicles, and trucks.

Object detection with no clutter   For this experiment, we use the StreetScenes data set and the experimental methodology carried out in section 3.3.2, wherein the three binary tasks are to discriminate between images of cars, pedestrians or bicycles, and background in a *crop-wise* detection task. The objects in the positive data are relatively standardized in terms of position and scale within the crops, and the background class consists of image crops known not to contain any of these object types. We should note that this is a difficult task owing to the large amount of

104

internal class variability. For instance, the car data set includes vehicles of many different types, from small coupes to large busses, and at many different poses, but they are all labeled as only one class. Example images were shown in Ch. 3. The data set is extracted from our StreetScenes database, as described in Ch. 2. Briefly, it includes 3,547 images with 5,001 labeled cars, 1,449 labeled pedestrians, and 209 labeled bicycles, among other objects. In the experiments, the data was randomly split into $\frac{2}{3}$ for training and $\frac{1}{3}$ for testing.

In chapter 3 it was shown that, for these three data sets, the C1 features outperform several other state-of-the-art feature sets, notably, the two systems described by Torralba and Leibe in [117] and [68]. We therefore compare only to the C1 features and to the similarly successful HoG feature set. As can be seen in table 5.1, the baseline systems' relative performance depends on the object.

Dataset	Car		Pedestrian		Bicycle	
measure	EER	tp@fp = .01	EER	tp@fp = .01	EER	tp@fp = .01
C1	5.62±1.1	81.73±4.1	18.41±2.2	32.83±6.1	8.57 ± 2.2	59.79±10.0
C1 + Cont	4.58±1.0	86.88±3.9	8.07±1.2	71.02±4.5	7.28±2.1	78.33±7.2
C1 + LinCont	4.87±1.0	86.72±4.0	10.14±1.7	56.33±6.4	8.51±2.3	66.31 ±8.0
C1 + Circ	4.68± 0.9	86.89±3.9	8.27±1.7	67.98±4.8	6.79 ± 2.1	79.36±5.5
C1 + LinCirc	4.92±1.0	85.55±3.8	9.79±1.3	61.90±5.8	8.27±2.2	68.26±9.9
C1 + Par	5.07±0.9	84.24±3.6	10.50±1.3	58.23±7.9	8.87±2.5	65.39±9.7
C1 + LinPar	4.87±1.0	86.72±4.0	10.14±1.7	56.33±6.4	8.51±2.4	66.31±8.0
C1 + Symm	4.92±0.9	86.01±3.4	9.00±1.5	57.05±5.2	6.37±1.9	76.52±9.0
C1 + All Four	3.6±0.7	90.9±2.6	4.8±1.0	85.2±3.2	6.2±2.6	84.7±6.7
HoG	8.62±1.1	61.36±6.7	9.81±1.5	62.62±6.5	12.14 ± 2.7	52.90±9.3
HoG + Cont.	6.93± 0.9	69.88±5.7	7.67±1.0	72.81±6.1	7.84±2.9	73.29±7.7
HoG + LinCont	6.69± 0.7	68.96±5.8	7.71±1.2	67.90±6.6	9.87 ± 3.2	65.44 ±10.7
HoG + Circ	5.82±1.1	75.11±4.7	6.66±1.4	76.84±5.1	6.99±1.8	76.14±8.1
HoG + LinCirc	7.06± 0.9	69.09±5.2	8.18±1.3	70.09±5.6	9.19±2.4	68.02±8.4
HoG + Par	6.78±1.0	73.83±5.7	7.49±1.5	71.75±5.8	9.87±3.2	65.44±10.7
HoG + LinPar	6.63±0.7	69.06±6.6	7.74±1.2	68.18±6.4	10.27±3.1	61.31 ±9.8
HoG + Symm	5.88±0.8	75.83±5.3	7.20±1.1	71.31±7.2	8.27±2.9	65.88±9.1
HoG + All Four	4.9± 0.9	81.9±4.4	6.4±1.2	79.4±4.3	6.4±1.9	79.0±7.9

Table 5.1: Object detection results for the Car, Pedestrian and Bicycle data sets using the crop-wise detection measure outlined in section 5.2.2. Each row indicates a feature-set. Each column lists ROC statistics in percentage: either equal-error-rate or true-positive-rate when the false-positive-rate is set to 1%. All statistics were generated by randomly splitting the data 25 times into training and testing sets. The "+ all Four" data is the results of a classifier trained on the baseline (C_1 or HoG), plus Continuity, Circularity, Symmetry, and Parallelism features.

There is significant improvement in performance when the continuity features are included in addition to the baseline features. In table 5.1 we list the performance of these classifiers both with and without the additional continuity features. The reported statistics were collected by randomly splitting the data into training and testing 25 times, training gentleBoost classifiers [56] until convergence, applying it to
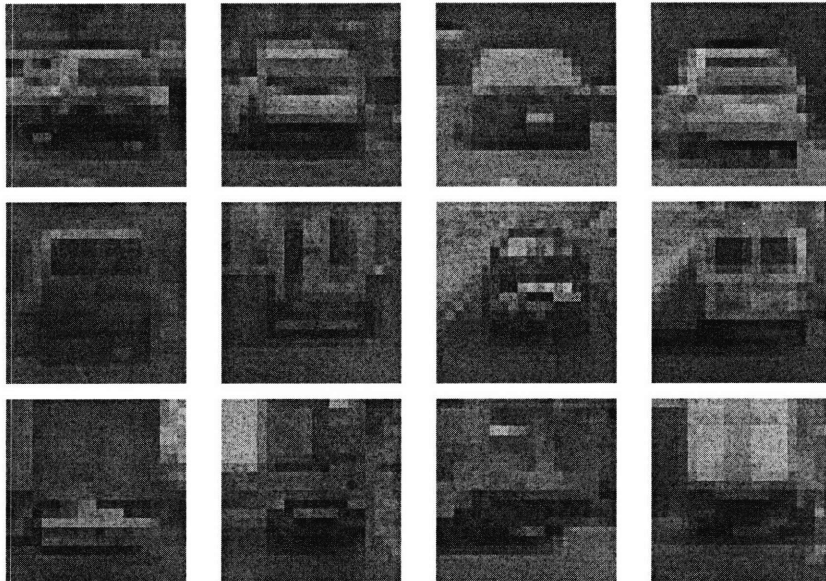
Figure 5-3: Twelve examples of private car (*top*) vs. mid-size vehicle (*middle*) vs. truck (*bototm*) images from the car database mentioned in Sec. 5.2.2. The task is difficult due to the very low resolution of the data and some ambiguity in the class distinction.

the test data, and then recording statistics of the resulting ROC curve. We report the equal-error rate and the true-positive-rate when the false-positive-rate $= 1\%$, since the low-false-positive region is the interesting part for the detection applications. It can be seen that the continuity features have significantly improved the power of the classifier.

Finally, to illustrate the necessity of the non-linear morphological operations in the continuity operator, we include the results of a more traditional linear version of the operator in the same table. This operator, instead of performing cycles of erosion, dilation, and image decimation, simply filters the oriented edge responses with analogous linear filters. It can be seen that this linear version helps the classifiers, but not nearly as much as the algorithm described in Fig. 5-1.

Car type identification The car type data set consists of 480 images of private cars, 248 of mid-sized vehicles (such as SUV's), and 195 images of trucks. All are 20×20 pixels, and were collected using an automatic car detector on a video stream taken from the front window of a moving car. The task is to classify the three types of

cars for a safety application. Taking into account the low resolution and the variability in the three classes, this is a difficult task (see Fig. 5-3).

The protocol used in the experiments is as follows. In each one of 20 repeats, 100 training images and 95 testing images were randomly drawn from each class. A multi-class SVM was trained, and the average success rate along with the standard deviation was reported. The gray-level features score 59.86% ± 3.35% on this three class problem, the $C1$ features score 41.89% ± 3.62% and the continuity features score 67.51% ± 2.92%. This success supports the postulation that classification is best performed on this dataset by using long horizontal and vertical edges. We also tried combining feature sets: gray-level combined with C1 scores 59.65% ± 4.24%, while gray-level and continuity together scores the highest score among our experiments; 72.35% ± 2.47%.

5.3   Circularity and Form based image descriptors

There is a well known result in studies the of statistics of natural images that, given the location of one oriented edge, the most likely locations for another oriented edge are those in which the two edges would have the property of *cocircularity* [43]. Similarly, in studies of saliency, it is known that humans tend to group together edge elements which form closed shapes such as circles or quadrilaterals [124]. In order to capture the notion of form in an explicit image feature, we have developed the algorithm detailed in Sec. 5.3.1. We will show that this mid-level image feature captures useful information about the image which is not immediately available in other state-of-the-art image representations.

5.3.1   The circle and form descriptor

In Figs. 5-4 and 5-5 we describe, in pseudo-code and diagrams, the algorithm used to calculate the circle feature. Each element in the vector output by this algorithm collects the evidence that there is a convex form (circle, rectangle or otherwise) of an approximate scale r centered at an approximate (x, y) location in the image. The

Given an input image I, apply the following procedure:

1. *Initial filter:* For each $\theta \in \{0°, 45°, 90°, 135°\}$, Let $D(x, y, \theta) = |\nabla_\theta(I)|$
2. *Non-maximal suppression:* if $D(x, y, \theta_1) < D(x, y, \theta_2)$: set $D(x, y, \theta_1)$ to 0
3. *Initialize 4-D voting space to zero:* $H(x, y, \theta, s) = 0$
4. *For each radius* $r \in \{2^1, 2^{1.5}, 2^2, 2^{2.5} \dots, 2^5\}$ *and orientation* θ:
    a. *Define an elongated gaussian shaped voting kernel:* $G_{(\theta,r)}$
       of size $2r \times 2r$ with a diagonal covariance matrix with values $3r$ and r.
    b. *Rotate:* $G_{(\theta,r)}$: by θ
    c. *Define the translation:* $t_x = r \times \sin(\theta)$, $t_y = r \times \cos(\theta)$
    d. *Collect votes by convolving the voting kernel and translated filter response:*
       $H(\mathbf{x}, \mathbf{y}, \theta, r) = H(\mathbf{x}, \mathbf{y}, \theta, r) + (\mathbf{G}_{(\theta,r)} \circledast D(\mathbf{x} - t_x, \mathbf{y} - t_y, \theta))$
       $H(\mathbf{x}, \mathbf{y}, \theta, r) = H(\mathbf{x}, \mathbf{y}, \theta, r) + (\mathbf{G}_{(\theta,r)} \circledast D(\mathbf{x} + t_x, \mathbf{y} + t_y, \theta))$
4. *Maximal suppression:*
       if $\forall \theta_2, H(x, y, \theta_1, r) \geq H(x, y, \theta_2, r)$ then set $H(x, y, \theta_1, r)$ to 0
5. *Sum the voting matrix H over orientations*
6. *Reduce image resolution:* using bilinear interpolation by a factor of 8

Figure 5-4: Algorithm to compute the circularity feature vector of an image.

algorithm presented here is in some ways similar to the generalized Hough transform used to detect circles, though there are also distinct differences. In the usual framework, boundary elements "vote" in a non-parametric way into the circle-parameter space. Similarly, in our system, each edge element spreads its vote, modulated by the edge magnitude, into data structure indexed by (r, x_c, y_c), the postulated circle's radius and center. However, since our edge elements contain orientation information, they only vote for circle-centers if they would be tangential that circle. A wide, elongated voting kernel is used in order to represent shapes which are not precisely circular and to prepare for the coarse descretization necessary to restrict the feature vector length to a manageable size. Finally, we use a non-standard step in order to remove noise from the result. In the usual Hough framework all edge elements vote equally into the parameter space. The natural extension would be to have all oriented edges contribute equally into the parameter space, according to their magnitude. Instead, for each point in the parameter space we remove all votes from the strongest contributing orientation. The effect is that only circles with support from more than one orientation are retained. Although they are not shown for reasons of space, results suggest that sum-minus-max outperforms other operators in its place: the sum,
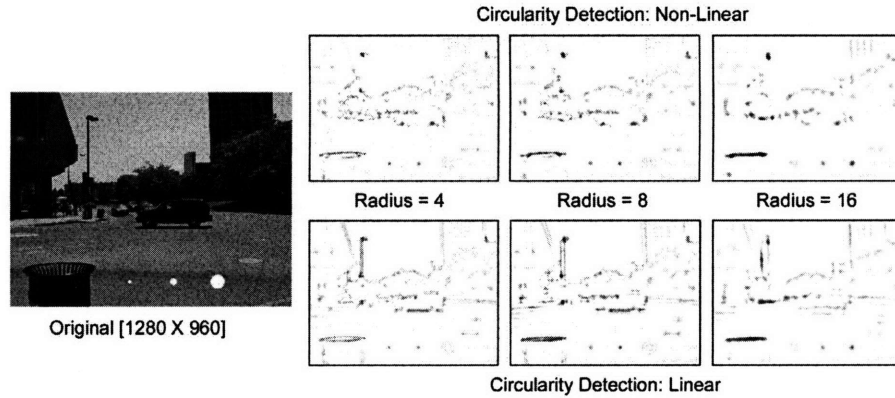
Figure 5-5: An illustration of the circle feature operating on a natural image (white circles of radius 4, 8, and 16 have been added artificially for illustrative purposes). In the bottom row, step 4 has been removed to make the computation more similar to the standard Hough transform. Note especially that the linear version gives strong circle detections over the straight edges in the **light post**, **building**, and **car shadow**. The non-linear version does not, and in the building only responds strongly to the rectangular windows at the appropriate scale. Note also that the detection of the **car tires** is obscured by the car edge detection in the linear version.

max, and median. Moreover, the summation of magnitudes greatly outperforms the common generalized Hough transform framework, i.e., thresholding followed by one vote per location.

Our current implementation of this circularity operator costs approximately one second per 128×128 pixel image. This is computing circles at 9 different scales. Each scale is computed independently, so the computational cost is linear in the number of scales represented.

Property 3 (elimination of common distractors) *Our algorithm (Fig. 5-4) reduces the support of imaginary circles generated by long straight boundaries in an image. See Fig. 5-5 for an illustration.*

5.3.2 Experimental validation

Again we demonstrate performance on the StreetScenes crop-wise detection task by appending the baseline features with the circularity features. As can be seen in table 5.1, for each object category, the inclusion of the circle features significantly improves
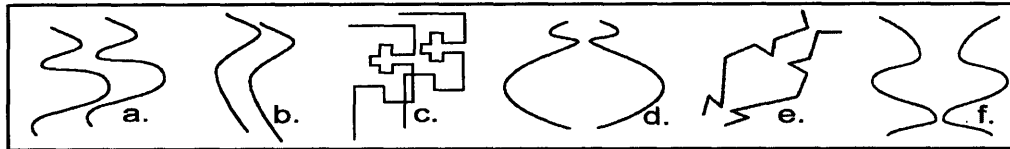
Figure 5-6: Examples of parallel (a–c) and symmetric (d–f) figures.

the detection score. Our experiments suggest that the choice of classifier, whether gentleBoost or SVM, is not important.

In order to demonstrate the importance of the non-linear *sum-minus-max* step, we also show the results of a similar algorithm which does not remove the support from the dominant orientation. The results of this comparison are listed in the same table under the heading, "circle linear." For all three classes tested, the non-linear circle representation is significantly more powerful. This likely due to the superfluous form detections caused by the prevalence of straight edges in the images, a feature which is already captured in the baseline representation.

5.4 Symmetry-based image descriptor

It has been shown that humans are extremely adept at detecting symmetry in natural images. As symmetry is a relatively rare phenomenon in images, it is likely that it is a useful cue in the detection of objects that exhibit symmetric characteristics. A measure of symmetry is defined at a position p, scale s, and orientation θ in an image. Strong symmetry at (p, s, θ) means that if we imagine a dividing line with orientation θ passing through p, the image on one side is in some way similar to a reflection of the image on the other side, at scale s. Previous object detection studies examining notions of symmetry include the work of Saber, Zielke, and Marola [97, 133, 76]. An illustration of symmetry of the type we will describe here is shown in figure 5-6

5.4.1 Calculating Symmetry as an Image Feature

The symmetry descriptor takes an input grayscale image and outputs an explicit measure of mirror symmetry for local regions within the image. Specifically, this

110

operator measures symmetry by looking for axes of *vertical mirror* symmetry. While we leave out many other types of symmetry, such as mirror symmetries at other axis, axial symmetry, etc., there is a great deal of evidence that humans are far more sensitive to this of symmetry than others [127]. Briefly, our symmetry operator will produce a feature vector in which each element estimates for a small area of possible axis locations, for a small area in the orthogonal y direction, and for a few different scales σ, how much symmetry evidence there is. The algorithm, in pseudo-code, is listed in Fig. 5-9. Furthermore, in Fig. 5-7, we show the results of the symmetry detection algorithm on some test images. It is easy to see that the symmetric structures are well represented by the feature.

The algorithm detects symmetry by accumulating evidence from pairs of matching patches. Each patch is compared to mirrored images of patches located across a posited line of symmetry. By taking the maximum of the match strength over a small pool of locations we build tolerance to variations in the depth of the symmetric object and small rotations of the symmetry axis. Afterwards, all match strengths which would contribute to the same line of symmetry at the same location are summed. Thus, two mirror patches near to the line of symmetry and a pair far from the line of symmetry both contribute to the symmetric stimulus. Finally, after accumulating evidences of symmetry at all points within the image and at a discreet set of scales, the data is reduced in such a way so as to retain the essential qualities of the symmetric information, but fit within a feature vector of reasonable size. This is done through bilinear resizing of the matrix after taking a local maximum, since we are not concerned with the exact pixel location of the line of symmetry so much as the fact that at least one strong symmetry signal exists within reasonably sized region.

The symmetry feature is very expensive to compute, requiring approximately 80 seconds per 128 $\times$ 128 image, using the implementation described here. This time can be cut down significantly by calculating the patch matching scores in grayscale, as opposed to a S_1 like oriented filter space, although this affects performance. The major cost is in computing the matching scores for all the patches, so it is likely that the speed can be improved greatly by using a form of approximation, such as PCA,

to rapidly compute the large matrix S.

5.4.2 Experimental validation

In Table 5.1 it is shown that the addition of the symmetry features to the baseline C_1 and HoG features enables the object detectors to achieve significantly better results on all three objects tested. No linear version of the symmetry detector was constructed, but it is likely that the maximum and minimum operations are important to the efficacy of the accumulation process which collects groups of patch matches into local symmetry measurements. A linear version, in which the max and min operations are replaced with linear filters, would be weaker at reducing the sparse symmetry detection matrix to a manageable size.

In order to compare the importance of the representation of the patches, we compared symmetry using gray-scale patch matching to the patch matching in oriented filter space, as described in the pseudo-code. Our experiments show that while grayscale matching is capable of capturing some of the notion of symmetry, as expected, the patch-matching in orientation space was more robust and led to greater performance improvements on the the detection task.

5.5 Parallelism based image descriptor

The notion of parallelism is similar to symmetry, but the parallelism measure does not require a reflection. Note that parallelism in terms of parallel geometric lines is but one form of the parallelism as we refer to it here. Fig. 5-6 illustrates the concept of parallelism as we refer to it here. Below we will describe an image feature to represent parallelism in images, and demonstrate its utility in object detection.

5.5.1 The parallelism descriptor

The algorithm to describe image parallelism is detailed in the pseudocode in Fig. 5-10. This feature is designed to capture whether or not there exists a consensus of
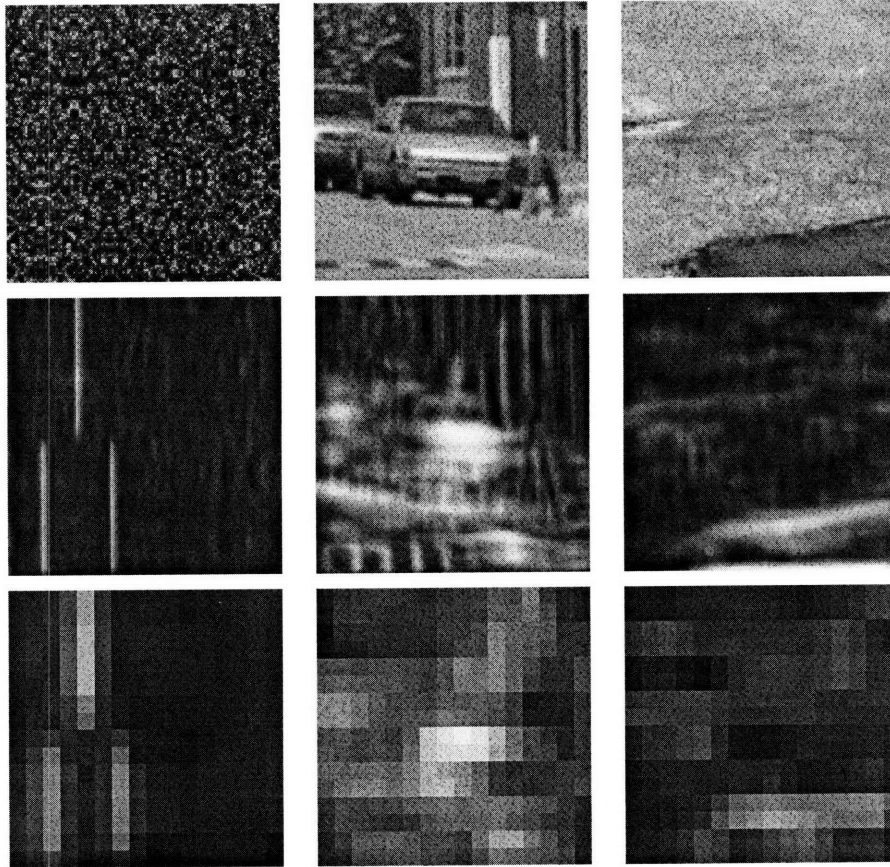
Figure 5-7: An illustration of the symmetry detection algorithm on three test images. In the symmetry response images, (*middle* and *bottom*), brightness indicates strength and the color indicates the scale of the detected symmetry, with blue representing smaller regions than red (white regions have symmetry detected at all scales). The leftmost toy image contains three regions of perfect vertical symmetry and three regions of perfect horizontal symmetry. The vertical symmetries are strongly detected in the response images. The symmetry of the car image is detected as a region of strong symmetry at the center of the image. As can be seen in the third column, random textures have little symmetry, and continuous lines have symmetry at a small scale, but not larger. The bottom row indicates the resolution at which the feature is used in the detection experiments.
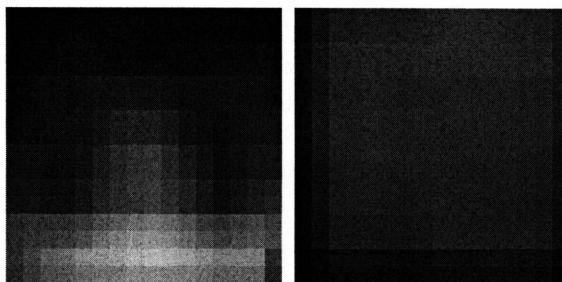
Figure 5-8: The left image indicates the average symmetry response for the cars in the StreetScenes database, while the right image shows the average symmetry response from the non-car data-set. Brighter pixels indicate on average stronger symmetry. The color indicates the scale of the symmetry, with blue indicating smaller symmetries than red. From these images we see that the symmetry descriptor is discriminative for the car class, as cars tend to have symmetry in the center and bottom of the image, whereas non-cars do not tend to exhibit any structured symmetry.

Given an input image I, perform the following algorithm

1 Initial filtering: Compute absolute derivatives of I in four orientations $\quad D(x, y, \theta)$

2 Loop over scales: For a discrete set of scales $\sigma \in \{2^{\frac{0}{3}}, 2^{\frac{1}{3}}, ..., 2^{\frac{15}{3}}\}$

  a: Let $\overline{D}$ equal D resized by a factor of σ in x and y $\qquad D \to \overline{D}$

  b: *Calculate Local similarities:* $\forall \{x, y, d_y, d_x\}$, $d_y \in \{-7...7\}$, $d_x \in \{0...25\}$

   b_i: Let P_1 be the $7 \times 7 \times 4$ patch of $\overline{D}$ centered at $(x - dx, y)$

   b_{ii}: Let P_2 be the $7 \times 7 \times 4$ patch of $\overline{D}$ centered at $(x + dx, y + dy)$

   b_{iii}: Let $S(x, y, d_x, d_y) = \text{NormCrossCorr}(P_1, \boldsymbol{mirror}(P_2))$ $\qquad S(x, y, d_x, d_y)$

  c: Half-wave rectify S $\qquad \max(S, 0)$

  d: Retain only maximum of S over d_y $\qquad S(x, y, d_x)$

  e: Take sum of S over all d_x $\qquad S(x, y)$

  f: Resize $S(x, y)$ to 128×128 and store as a layer of G $\qquad G(x, y, \sigma)$

3 Reduce feature vector length:

  a: Local min of $G(x, y, \sigma)$ with a kernel of size 8 in y $\qquad \ominus(1, 8, 1)$

  b: Local max of $G(x, y, \sigma)$ with a kernel of size 8 in x $\qquad \oplus(8, 1, 1)$

  c: Perform a 8×8 subsampling in (x, y) $\qquad \downarrow (8, 8, 1)$

  d: Bilinearly resize $G(x, y, \sigma)$ to $16 \times 16 \times 4$ $\qquad \Downarrow$

4. $G(x, y, \sigma)$ is our 128 dimensional feature vector. $\qquad G(x, y, \sigma)$

mirror:

Given a multi-layer image $I(x, y, \theta)$ where x ranges from 0 to x_{max}, and θ ranges from 0 to 180, the mirror image $\hat{I}$ is defined as $I(x_{max} - x, y, 180 - \theta)$.

Figure 5-9: Algorithm for computing image symmetry features. A feature vector of reasonable length is computed from an image of arbitrary size.

Given an input image I, perform the following steps:	
1. *Initial filtering:* Compute absolute derivatives of I in four orientations.	$D(x, y, \theta)$
2. *Local similarities:* Compute local similarities of every $5 \times 5 \times 4$ image patch to all neighborhood patches of a distance $4 \leq \{d_x, d_y\} \leq 25$.	$S(x, y, d_x, d_y)$
3. *Perform the following sequence of morphological operators:*	
a. Local maximization with a kernel of size 3×3 in (d_x, d_y)	$\oplus(1, 1, 3, 3)$
c. Perform a 3×3 subsampling in (d_x, d_y) .	$\downarrow(1, 1, 3, 3)$
b. Local summation with a kernel of size 16×16 in (x, y).	$\sum(16, 16, 1, 1)$
c. Bilinearly resize S by a factor of $\frac{1}{8}$ in (x, y) .	$\Downarrow(8, 8, 1, 1)$
d. Local maximization with a kernel of size 7×7 in (d_x, d_y).	$\oplus(1, 1, 7, 7)$
e. Perform a 7×7 subsampling in (d_x, d_y) .	$\downarrow(1, 1, 7, 7)$
4. *Obtain a 4D result.*	$R(x, y, d_x, d_y)$

Figure 5-10: Algorithm for computing image parallelism features.

relative locus of similarity within spatial sub-regions of the image, i.e., if a cluster of pixels all agree that they are similar to a certain shifted sub-region of the image. First, the image is filtered and rectified into a set of 4 orientation images. In stage 2, each $(5 \times 5 \times 4)$ image patch is compared to all patches of a similar size within its neighborhood. L_2 distances of the edge-response patches is used as the similarity measure so that edges only match similar magnitude edges of the same orientation. In stage 3 the local maximum of this similarity measure within a (3×3) region of the relative transformation is taken order to allow a small amount of distortion in the parallel shape. Next, for each relative location separately, a local sum is taken in the space of the image plane. If a group of nearby patches agree that the image is similar at this relative location, then the sum will produce a peak at the center of the group.

Finally the data structure is processed via local information pooling and subsampled. This is done for two reasons, firstly to reduce the feature vector to a manageable size for modern statistical learning methods, and secondly to allow tolerance to the exact location and nature of the parallelness. The responses in the (x, y) image plane are pooled by adding them locally, in order to compute a consensus of similarity, but the responses in the (d_x, d_y) direction are pooled by taking the maximum. The intuition behind this is that we are only interested in the one translation that gives the best explanation of parallelness, not in the average parallelness over all translations.

The total computation cost for the Gestalt parallelness feature over an image of

128×128 pixels is approximately 60 seconds. This cost is dominated by the need to calculate the local patch correlations for all locations within the image and over a set of translations, d_x and d_y. As in symmetry, it is likely that the feature can be computed much faster using an approximation. Also, we should note that our method is certainly not the only method of computing parallelism. It might be the case that some other method is able to extract the parallelism information equally well and with much lower computational cost.

5.5.2 Experimental validation

Referring once again to table 5.1, we see that the addition of the parallelism features enables the object detectors to achieve significantly better results on all three object databases. In order to demonstrate the necessity of the non-linearity in the computation of the feature vector, we have produced a second version of the parallelness detection algorithm, in this case replacing all of the dilation operations with sums. We see again that non-linearity is an important part of the computation, but this is not as consistent across datasets as in the other descriptors. Note that if the dilation operations are linearized, then the entire algorithm can be collapsed into one sum on the original data structure, i.e., each feature would simply be the sum of a set of individual patch similarities.

5.6 Experiments on the 101 objects dataset

A final supporting result is given on the popular 101 objects dataset [37]. The results reported here are the average and standard deviation, taken over all 101 classes plus the background class, of the object recognition performance obtained from 20 independent trials. In each trial 15 random training and 50 random testing images were selected from each class, fewer testing images were used if not enough were in the database. In the final score, all errors are weighted such that each class has the same contribution regardless of the number of the testing images. Note that by using all of the testing images at once, one gets a much better performance, due to the

116

over-representation of some classes. Therefore, to compare with previous results, we use the 15/50 protocol with reweighting.

Using this protocol, Berg et al.[7] report 45% correct detections on the non-duplicated version of the dataset. Serre et al. [103], report an average performance of 35% using 10,000 "global" standard model C2 features, but using the publicly available code, combined with a per-feature variance normalization step, we were able to improve this performance to $36.86\% \pm 1.64\%$ using only 3,000 such features. (All of the results were obtained using a one vs. all linear SVM).

Using the same splits, the C1 features gave $30.93\% \pm 1.20\%$ by themselves, and $44.18\% \pm 0.76\%$ when added to the 3,000 C2 features. Continuity gave, when combined with C2 $44.59\% \pm 0.93\%$, and 30.45 ± 0.34 by itself. Circularity gave, combined with C2 $41.48\% \pm 0.52\%$ (26.96 ± 0.88 alone). Parallelism by itself received $17.10\% \pm 0.35\%$, and did not help C2 (37.47 ± 1.05). All the features taken together C2, C1, Continuity, Circularity and Parallelism produced a score of $48.26\% \pm 0.91$, which is the highest result ever reported. Symmetry was not available at the time of this test.

Note that, although Continuity, Circularity and Parallelism did not score very high individually, they do outperform the results of Fei-Fei et al. (16% [37]) and the baseline result of Berg et al. (17% [7]). The main contribution of these features, however, is not in representing an image by themselves, but as a supporting feature. The additional gain over a system which already performs well is valuable, by the law of diminishing returns.

5.7 Random Visual Features

It is a valid criticism that the Gestalt features are highly engineered, and there is little support that their complexity is warranted. Is it possible that any image feature made of the same sorts of elements can lend equal support to the detection problem? We have attempted to address this problem by building *random* image features and showing that the specially engineered features are better than random features built

Operation	Parameters	Possible Parameter Values
Dilation	Kernel	Any (5 × 5) binary matrix approximating an oriented bar. Thickness and orientation are drawn independently from $\{0°..180°\},\{1..7\}$.
Erosion	Kernel	Same as above.
Abs(Linear Filter)	Kernel	either (50%) (5 × 5) Gaussian of random Σ or (50%) (5 × 5) Gabor of random orientation.
Bilinear Resizing	Newsize	$\{0.25$ to 0.85 of current size$\}$, drawn independently for each dimension.
	Method	either bilinear or nearest neighbor.
Decimation	Dimension	either x or y.
	Span	any integer in $\{1..8\}$.

Table 5.2: The atomic operations defining the space of random features. Each random feature was constructed by building an independent random sequence of these operations, and selecting random parameter values for each operation.

of similar components.

In order to select the random features, it was necessary to posit some space to randomly chose them from. In order to make them similar to the gestalt features described above, it was decided to begin with the image filtered with four oriented filters, and then to randomly select a *sequence* from a familiar set of operations. This set included linear filters, local maximization, local minimization, down-sampling, and bilinear resizing. Random selections are made from this set until the entire feature length would be less than $1,000$ values. For each atomic operation, many parameters are possible, such as the type of filter, the dimension and span with which to down sample, etc. These parameters were also selected randomly from a predefined set designed to mimic the types of oriented Gabor filters, Gaussians, and other tools used in the engineered features. For instance, the local maximization can take place over a local rectangular field, or an oriented bar, like in the continuity operation. The random operations are listed in table 5.2.

Ten such random feature-computations were generated. In order to save time, rather than testing on all three object types, only cars were tested in this experiment. Similarly to previous experiments, the new feature was concatenated with the baseline

feature, either of C_1 or *Triggs*, and 25 trials were run using random train-test splits. The results of these experiments are illustrated in figure 5.7. We see that the addition of these features does tend to improve performance slightly over the baseline levels, but not to the same extent as our engineered features. It is also interesting that some features seem to be significantly more useful than others. Perhaps an even stronger classifier can be built by searching the space reachable by these features using some searching algorithm, such as genetic algorithms (GA). It is probably beyond our current computational power to do such a search, however, as we would need to perform many tests on many different types of objects in order to get an idea of the real value of the feature without over-fitting to one particular data set. These computational issues were addressed in previous GA-for-image-features studies by limiting the search to either a small feature-computation space or tuning on only one object database [59, 99].

As an aside, it is conceivable that by descretizing the set of parameters for each atomic image operation one could even build a countably infinite set of the features and search through them in order. The set of features described here includes, as points in the space, the SIFT operation, HoG modulo the normalization step, C_1, and the continuity feature described above. By adding or removing computational atoms, one can adjust the set of possible features. By allowing offsets it would be possible to build the Gestalt circle feature, and by adding offset-autocorrelation one could construct the parallelism and symmetry features. The same few building blocks, when organized in different patterns, can perform a wide variety of disparate image operations.

5.8   Parameter Tuning

In the design of our gestalt features we have taken the liberty to engineer solutions which capture the target gestalt image properties. These solutions have often included chosen parameters which have no formal rationale for their hand-selected values. In this short section we show a few scans across relevant parameters showing how the
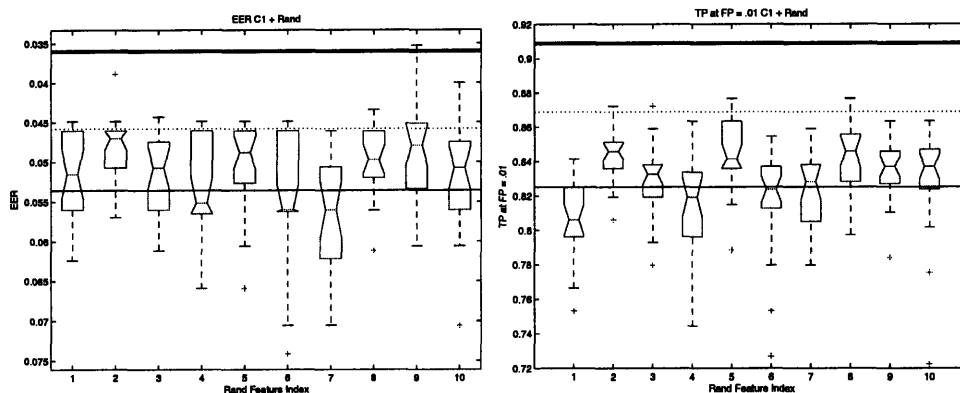
Figure 5-11: Performance of the randomly generated image features on the crop-wise car detection task. Ten random image feature algorithms were generated and tested on the car data. The resulting features were concatenated with the C_1 data and tested similarly to the engineered Gestalt features. The two box-plots above display the performance in terms of equal-error-rate (*left*) and true-positive-rate when false-positive-rate equals 1% (*right*). Note that the y-axis on the left graph has been flipped to make the results perceptually similar, up is better. Superimposed on the box plots are three horizontal lines. These lines correspond to the performance of the C_1 features alone (*thin line*), the C_1 features plus the continuity features (*dotted line*), and the C_1 features plus all four gestalt features (*thick line*). As most results lie between the thin line and the dotted line, we theorize that the randomly generated image features are helpful to the detection problem, on average, but not as helpful as the continuity features. Results adding random features to HoG are similar.

120

features' utility can depend on specific parameters chosen.

5.8.1   Kernel Size in the Continuity Computation

Referring to Fig. 5-1, the pseudocode describing the continuity algorithm, in steps $3a$ and $3b$ a morphological operation is performed using a kernel of a specified size. This kernel size was selected so as to be large enough to capture a significant amount of information about the local support for the continuous edge, but not so large as to be computationally inefficient. Furthermore, the relative size of these two kernels significantly changes the properties of the algorithm. If the dilation kernel is larger, then gaps in the edge can be bridged, but hallucinatory contours are often detected in textured regions. Conversely, if the erosion contour is larger then only true contours are detected, but small gaps, often due to noise, grow to consume the contour at some point.

In two separate crop-wise car-detection experiments, we vary the size of these kernels and demonstrate that the performance gain is relatively tolerant to variation therein. In figure 5-12 the diameter of the dilation kernel is varied while holding the erosion kernel constant at 7 pixels. The best performance is when the size of the two kernels is matched, with performance dropping off when the dilation kernel is too long or too short. In another experiment we varied the size of the two kernels simultaneously, but the results showed only that the feature's utility is relatively tolerant to this parameter.

5.9   Conclusion

The idea that more meaningful image representations can produce significantly better recognition results is attractive, but it is not trivial to demonstrate. In this chapter, we have revisited the principles that were used to build effective histograms-of-orientations-based features, and used them to derive interesting mid-level features. Histograms consisting of spatial bins are used, just like in SIFT and HoG, and non-linearities are employed in a way which is not unlike the maximization in C_1. Patch
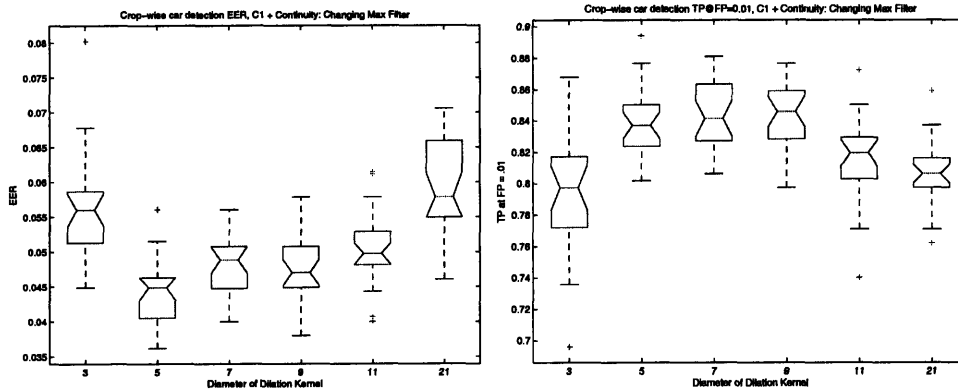
121

Figure 5-12: Variations in the size of the local maximum kernel in the continuity algorithm (stage 3b) effect the performance on a crop-wise car detection task. Here we only show performance of a classifier trained on C_1 plus continuity, but results on HoG plus continuity are similar. The hand-selected value for this parameter, 7 pixels, is close to the local optimum performance, which seems to depend on which performance measure one chooses to optimize for.

based approaches in the parallelism and symmetry operators are similar to their function in C_2. By using these atomic operations, and changing the details, we have built new image operators enabling us to describe useful mid-level concepts.

The novel features presented in this chapter collect image information that is not necessarily concentrated in space (HoG, SIFT, C_1) or in scale (C_1), but instead along other modes of image structure. Evidence pooled from image gradients along lines or circular arrangements is combined to support hypotheses of image structures that are unlikely to be coincidental. Patch-based similarities are grouped based on coincidence the spatial patterns of similarity in order to build an effective representation of parallelness and symmetry. Other Gestalt features, such as grouping or $3 - D$ form may constructed using similar structures. It should be noted that our objective was simply to represent these concepts in any way to see if they could assist the object detectors. It is likely that there other, perhaps better ways of extracting this Gestalt information. Our goal was simply to show that this information is useful.

The four novel image statistics improve substantially the performance of state-of-the-art detectors. The performance gain is consistent across several challenging real world datasets, indicating that the framework is fruitful.

After noting the increased performance due to the gestalt features, a natural question to ask was whether these types of features might be computed in the human anatomy. Looking into the psychophysical research on symmetry perception, it was decided to repeat a recent experiment by Pashler [54], in which human subjects were asked to discriminate symmetric from near-symmetric stimuli, and to discriminate near-symmetric from non-symmetric stimuli. We trained classifiers were trained to perform the same task using either the C_1 features or the Symmetry features described in section 5.4. The symmetry features drastically outperformed the C_1 features on this task, but this was expected. What was surprising was how closely the C_1 results matched the human results. This suggests there may not be any dedicated symmetry processing in the feed-forward pathway in the human anatomy.

In cooperation with the lab of Olvia, we are in the process of running some psychophysical studies using rapid presentation tasks to see if the standard model is enough to capture humans' ability to recognize all these Gestalt stimuli, or whether the additional gestalt features are necessary. We look forward to exploring which, if any, of these features might be computed in the biology.

Chapter 6

Feedback Strategies for Hierarchal Feature Models

To date, there is no satisfactory explanation for the role of feedback connections in the primate visual system. These abundant feedback connections dominate the visual areas in a 10 to 1 ratio over the feed-forward connections [21]. A clearer understanding of their function could give insights into how to build better *computer* vision systems.

Theories for the role of feedback connections have usually focused on attentional mechanisms and hypothesis verification loops. In Hawkins [50], it is proposed that feedback connections are involved in prediction-verification recursions wherein the generation of a predicted representation (top-down) is matched with the upcoming representation (bottom-up). Computational implementations of these hypothesis-verification theories have had limited success when compared to appearance-based recognition systems, although some exceptions exist (*e.g.*, [17]).

In Reverse Hierarchy Theory (RHT), proposed by Hochstein and Ahissar [53], information is first processed automatically bottom-up, while perception progresses in the opposite direction, with conscious access to generalities first, and details later. RHT can account for many seemingly disparate psychological phenomena found in humans, such as the different modes of visual search, and the properties of vision at a glance compared to vision with scrutiny. However, in biological systems, verifying this or any other feedback theory remains extremely difficult.

As far as synthetic computational systems, feedback is often used in the training phase of neural networks and graphical models. However, the authors are unaware of any work showing how feedback can be useful for modern *discriminative* object recognition systems, such as those employing SVM or AdaBoost for classification. Currently, these discriminative systems seem to outperform other technologies but still fall far behind human performance. Adding feedback is one promising direction to close the gap.

Here we create computational implementations of different hierarchal architectures to see if an improvement can be made over the performance of simpler discriminative object classification systems. In our framework, we equate the cognitive notion of perception of an object (from RHT) with the computational output of classifiers trained to respond strongly to that object. We describe five computational hierarchies and experimentally evaluate their relative performance.

6.1 Background: Hierarchical vision systems

Before appearance-based models became standard practice, hierarchal representations were common in computer vision. Objects were often represented as compositional hierarchies of atomic forms, and object searches were performed by detecting for appropriate groups of these atoms in oftentimes combinatorial sized search spaces (*e.g.*, [77, 32]). With the advent of advanced statistical learning tools and more powerful computers, the best performances were next reported by shallow architectures using learning directly on top of relatively simple image representations. Recently, hierarchal systems, now with multiple levels of learning, have regained their popularity (*e.g.*, [65, 101]).

Along with the additional power and flexibility available to hierarchal systems come problems inherent to their design. It remains unknown how best to factor the vision problem: should object parts be represented explicitly, at what stages should learning be employed, etc. In [118], a generative model was trained using both the appearance of target objects, and the gist of the scenes in which these objects were

likely to appear. While factoring the conditional of a generative model into layers of hierarchy is a well defined problem, the best way to combine these layers in a *discriminative* framework is less clear.

Part-based systems These are hierarchal architectures which explicitly represent spatially localized structures related to the task. For instance, the part-based face detector detailed by Heisele et. al. [51] employs detectors for the eyes, nose and mouth. The part based system of Ullman et. al. [121] is similar in that sub-parts of the image are represented explicitly by independent detectors within the system. However, in this system, instead of hand selection, mutual information is used to select which object parts will be represented. Other part-based object detection systems which automatically learn, via clustering or otherwise, which object parts to explicitly represent include [128] and [67]. In [13], feedback is used in order to resolve ambiguities in the part detections by considering inter-part relations. These part detections subsequently become the input to a higher decision making stage.

Classifiers as features Several systems have used discriminative classifier outputs as learned intermediate visual features, *e.g.*, [113, 5]. In this case, rather than detecting related parts of the object, the detectors are tuned to respond to all sorts of possibly unrelated objects, *e.g.*, one might imagine using the output of a pedestrian detector as a feature when building a building or horse detector. The intuition is that classifiers trained to perform tasks in one domain may learn to implicitly represent patterns and invariances in a host of related domains. The subsequent use of these classifier outputs as input to a higher level of the hierarchy allows for the transfer *domain knowledge*. Another related problem is to learn an intermediate level image representation without the luxury of label information for the non-target-object categories. In this case, it may be necessary to use an unsupervised algorithm, such as clustering or parametric models, as in [67, 63] or [128].

The convolutional network The system of LeCun et. al. [65] is an example of a multi-layered neural network designed and trained via back-propagation to perform

object detection and recognition tasks. It can be seen as a type of visual hierarchy in which learning occurs at each level. Rather than explicitly representing parts, a hierarchal object representation is induced through the clever network architecture. Although feedback is used during the training phase, during testing the convolutional network is completely feed-forward, leading to very rapid performance.

The standard model As described in detail in Ch. 3, the standard model describes a quantitative model of the feed-forward pathway of the ventral stream in visual cortex. So far, the modeling effort has focused on the feed-forward architecture of the visual stream. Strong evidence suggests that feed-forward pathways are responsible for the first 100 msec. of biological visual perception [88, 112] and that the basic steps for recognition are completed in this time. Of course, a complete model of vertebrate vision must take into account image sequences, as well as top-down signals, attentional effects and the structures mediating them (*e.g.*, the extensive back-projections present throughout cortex).

Reverse Hierarchies Hochstein and Ahissar's Reverse Hierarchy Theory, (RHT) [53] proposes that visual information initially travels through the feed-forward visual hierarchy, and that perception begins at the higher levels, reaching the lower areas via feedback connections, forming a reverse hierarchy. They point to three lines of evidence to support this theory. Firstly, the gist of a scene, consisting of broad category information, is consciously perceived first, and detailed information is only consciously perceived later when vision with scrutiny is engaged. The ability to perceive the contents of rapidly presented images, as seen in the Rapid Serial Visual Presentation (RSVP) paradigm [90], as seen in repetition blindness and change blindness phenomena, is further support. Since the receptive field sizes needed to process the gist of a scene are only seen in neurons of higher visual areas, while cells coding for precise details are found in lower visual areas, these phenomena suggest that perception starts at the highest levels and travels backward to the lower levels. The second piece of evidence comes from the visual search research, where it has

128

been well established that there are two modes of search, one mode that is rapid and parallel and one mode that is slower and serial [120]. The rapid parallel detection of objects often uses complex features such as circles and faces that are only available in higher visual areas, suggesting again that high level areas have consciousness first [114]. The final line of evidence comes from visual learning research, where it has been shown that visual tasks that are learned more rapidly tend to generalize to other similar spatial stimulus conditions, while harder tasks that take a long time to learn tend not to generalize outside of the specific conditions where they are learned [36]. Hochstein and Ahissar hypothesize that learning that has greater generalization occurs in neurons with large receptive fields and thus must be occurring in higher level areas, and since this generalizable learning occurs earlier than specific learning, the authors conclude that learning occurs in higher areas first.

6.2   Alternative classification strategies

Given a hierarchical image representation, there are several alternative ways it can be classified. If each level of representation is given as a vector, one can concatenate all of the representations to a single long vector. Another option is to classify each level separately and then combine the classification results. Below we will name and describe five such options. In our experiments we focus on two level hierarchies since modern representations with more than two levels are rare. It is straight-forward to enumerate more strategies for hierarchies with three or more levels, and to generalize the structures we have defined to these larger hierarchies. Below, we clarify some of the basic concepts we use.

High level vs. low level   Strictly speaking, representation B will be higher in the hierarchy than representation A if computing B requires computing A. When processing takes place information is lost. Hence, it is always the case that higher levels of hierarchies are less informative than lower levels. However, these representations are often more explicit, resulting in a more straightforward classification. On the other
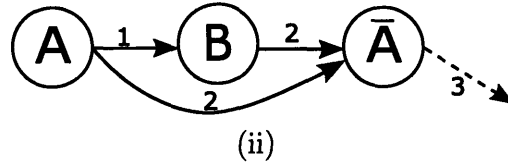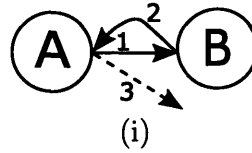
Figure 6-1: (i) A simple feedback system. (ii) The unfolded version of the same system. Since we are not concerned with the actual location of the computation we view system (i) and system (ii) as being equivalent.

hand, simply concatenating several representations will always result in no less information than any individual representation. Yet, it is the ease in which the classifier can access the *relevant* information that determines the overall accuracy, and having many irrelevant variables can diminish performance.

Feedback   In a *simple feedforward system*, each level of the hierarchy is used only to produce the next level. Once level $l - 1$ produces level l, its role is completed, and it is never used again. In systems that employ feedback, information from higher levels is passed back to lower levels and the combination of both is then being considered.

An example for a feedback system might be a system where representation A produces image representation B, which in turn through some process updates representation A. Finally, further processing is applied and an output is produced (see figure 6-1(i)). In this work we do not consider the location in which the feedback takes place. Instead we separate the initial representation (A) from the later representation ($\bar{A}$). Our view of the system would resemble figure 6-1(ii): representation A is computed, and representation B is computed from it. Representation $\bar{A}$ is then computed from A and from B, more processing is then done to produce the final system's output.

We assume that the feedback system and the unfolded system are equivalent. We describe our systems as unrolled systems, which resemble feed-forward systems.

130

(a) basic   (b) concatenation   (c) feed-forward perception

(d) reverse hierarchy   (e) semantic concatenation
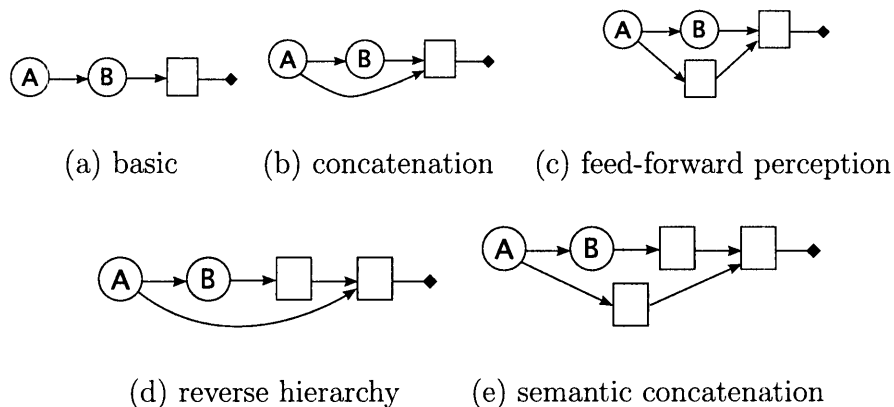
Figure 6-2: These figures illustrate alternative hierarchy architectures (this is not a full list). A circle represents an image representations, a square represents classification, the small full triangle represents the output. For example, (A) can refer to the C_1 image representation, while (B) can refer to the higher level C_2, a square can represent the output (raw distances from hyperplanes) of a 102 one-vs-all linear SVM classifier trained on the 101 objects (plus background) dataset. (a) Basic feedforward architecture. (b) Concatenation of the layers followed by a classifier. (c) A feedforward perception architecture in which the perception of the lower level representation is used in combination with the higher level representation to create the final perception. (d) Our interpretation of the reverse hierarchies [53]. The perception of the high level features is classified together with the low level features to create the final perception. (e) The semantic concatenation architecture, in which the final perception is a result of combining the perceptions of all of the previous levels together.

However, these system are not simple feed-forward systems (as defined above), and since they can be implemented as feedback systems, they can be studied as a model of these.

One can imagine the image representation $\bar{A}$ to be similar in nature to A, yet improved by incorporating higher level information that is available in representation B. For example, A might contain edge information, and by employing higher level information that is available in representation B, closed contours are being emphasized in $\bar{A}$. In a more elaborate example, B is classified (e.g., in a car detection system to a car or not a car) and the results of this classification are used to emphasize particular properties in A (such as the object boundaries or the removal of shadows in the car example).

We have conducted experiments akin to these two examples. Unfortunately, the results only demonstrated the difficulties in implementing a robust feedback system. In one experiment we tried to emphasize elements of the C_1 image representation [103] that contribute to long continuous boundaries as detected by our continuity detector [15]. The hope was that the modified C_1 would have better recognition capabilities than the original C_1. In another experiment, we tried to emphasize the most informative C_1 elements in a car detection task using feedback. For this experiment, we used a boosting classifier on-top of weak classifiers, each corresponding to a single C_1 unit. The units which contributed positively to the classification were made larger in value and projected back into the S_1 layer. Finally, a new C_1 representation was then computed. Both of these experiments produced disappointing results: the modified C_1 did not significantly outperform the original. Perhaps, with further study better results could be achieved.

In this paper, we are much more restricted in the kind of systems we allow. We assume that the modified representation $\bar{A}$ is to be delivered to a classifier. Since we do not know how to properly modify A to create $\bar{A}$, we build a classifier directly on top of the inputs that contribute to $\bar{A}$. Hence, if $\bar{A}$ is computed based on A and B, instead of classifying $\bar{A}$, we train a classifier directly on top of a combination of A and B. Since our classifiers are limited in power, and since the number of training

examples is not high, such a solution is not optimal. However, it does bypass the need to engineer a modified representation explicitly, which is a task for which our technological capabilities are limited.

Perception Another simplification that we employ is the equating of perception with classifier-output. Since perception is computed from the data it is worthwhile discriminating between processing (going up the representation hierarchies) and perception. In our terminology, perception is a type of computation which is learned from the data and which produces information which is aimed at being *directly* indicative of the objects' identity. In other words, through training, we expect the perception vector to be very highly correlated with the label information. In figure 6-2, representations are marked as circles, while perceptions are marked as boxes.

For multiple class data sets, we employ one vs. all architectures and the perception vector is as long as the number of classes. Note that we record the raw classifier output (a real number) rather than a discrete label. This "perception" can then be combined with other perceptions or with vector image representations simply by concatenation. For binary tasks, we create many classifiers, each using only part of the training data, and use the output of all of these classifiers to create a long perception vector (somewhat similarly to random forests [20]).

A crucial point is that for some hierarchies we train a first set of classifiers to generate perceptions and then train another set of classifiers on top of the output of the first set. Ideally, we would use one set of training examples (a validation set) to compute the first set of classifiers, and another set of training examples to compute the second set. In our experiments we do not use a separate training set for the following reasons: (1) when data are expensive, splitting the training set is wasteful; (2) Since we try to check if classification of classifiers' output is beneficial on a separate test set, the diminished performance one gets by using the same training data twice, can only lead to a more conservative conclusion; (3) We use classifiers with good generalization ability, limiting the effect of reusing training data.

6.2.1 Five alternative strategies

Below we present five alternatives strategies for classifying the information in a hierarchical representation. These alternatives are illustrated in figure 6-2.

basic (a) The most basic strategy to classify a hierarchical image representation is to simply classify one of the levels of the hierarchy. Usually, one would classify the highest level available [103]. Another alternative is to use the level in the hierarchy that produces the best results.

concatenation (b) Another simple strategy is to concatenate the features from all of the layers into one long vector, and use this vector for classification. This is the method we used in Ch. 5 and it seems to be quite effective.

feedforward perception (c) By feedforward perception we simply mean that the *perception* of the low level image description is concatenated with the *raw* high-level description to generate the final classification. As an example, consider a face detection problem in which we are given both the low-level pixel values, as well as a higher-level indications of the confidences of detecting certain face parts within the image. A feed-forward architecture might combine these two cues (pixel levels and part detections), by training one classifier directly on the low-level input, and then training a second classifier on the combiniation of the output of this classifier, and the part confidences. We consider this a feed-foreward architecture because the low level cues are classified directly to the object label first, building the low-level perception, and then this perception is combined with the higher-level cues, and classified again to build the final perception.

reverse hierarchy (d) RHT asserts that higher level representations are being perceived before low level ones. The initial perception is based on the rough high level information (or gist). This perception is verified while details are filled in by a second perception process that takes into account the initial perception and the low level representation.

In our experiments, we concentrate on classification accuracy, and not on the fine details of recognizing the object (for example, we do not try to recognize the object's boundaries or the parts of the object). This enables us to establish common performance measurements between the varying strategies. The high level feature is being classified to create a perception vector which is concatenated to the low level image representation and then classified to build the final output

semantic concatenation (e)   This strategy is usually referred to as stacking [132]. Here, perception vectors are computed for *each* level of the hierarchy separately, and are then combined by concatenation. The concatenated vector is classified to produce the final output.

This simple strategy resembles voting, but with important differences. First, while simple voting cannot take place between two classifiers, semantic concatenation is successful in such cases. Second, by our definition of perception, the classifier does not output a label but rather a real valued number. Similar to some voting techniques, in the final decision each element of the perception vectors may be weighted differently.

6.3   Experimental study

We cannot offer a theory that can predict which hierarchy is the most appropriate for any given task. Similar to situations where one has to choose between learning algorithms/parameters, performance measures on a validation data set might be the prevailing indicator. Below we provide experimental results on several vision data sets. These experiments show that the type of hierarchy used may have a large effect on the resulting accuracy.

In the experiments reported, when combining feature matrices X_1 and X_2, we normalize matrix X_2 by multiplying it by a constant such that the matrix norms of the kernel matrices $K_1 = X_1^\top X_1$ and $K_2 = X_2^\top X_2$ are the same. This way we ensure that their contribution to the final kernel used by the SVM, $K = K_1 + K_2$, is the same. Hence, even if $X1$ contains 3000 rows, each being a different C_2 feature, and

Method		Recognition rate
Edge	(a)	32.67 ± 1.1
Distance Map	(a)	35.25 ± 1.7
Both concatenated	(b)	48.33 ± 1.9
Feedforward perception	(c)	39.37 ± 1.5
Reverse hierarchy	(d)	34.50 ± 1.6
Semantic concatenation	(e)	51.37 ± 1.7

Table 6.1: Recognition rates in percent for several hierarchical strategies applied to the polygon dataset. The mean and standard-deviation were computed over 20 independent trials. The distance map representation is preferable to the edge map. Combining them provides the best results, but these vary depending on the combination method. Semantic concatenation (e) does best, followed by simple concatenation (b). Methods (c) and (d) do not perform well.

$X2$ contains 102 rows that are the output of one vs. all classifiers, their contribution would be similar. This normalization is irrelevant for the boosting classifiers, since the transformation is monotonic.

6.3.1 Toy data set – polygon identification

The random process described in figure 6-3 is used to create a data set of polygons with three to ten vertices. The goal is to classify each polygon image by the number of the vertices. Since the vertex location is random and the images are decimated down to 28 × 28 pixels, this is not a simple task to solve using an appearance-based model. For example, it is hard to distinguish between polygons of 8-10 vertices.

For this experiment, the edge image is used as the low level feature, and the distance transform is used as the high level image representation. The edge image was computed by applying the canny edge detector [22]. The distance transform is computed based on the edge image, and is clearly higher in the computational hierarchy.

The results are given in table 6.1. The success rates are averaged over 20 repeats of random experiments. For each experiment, 70 random images per class were used for training (a total of 560 training images) and 130 random images per class were used for testing. Random guessing would result in recognition rates of 12.5%

136
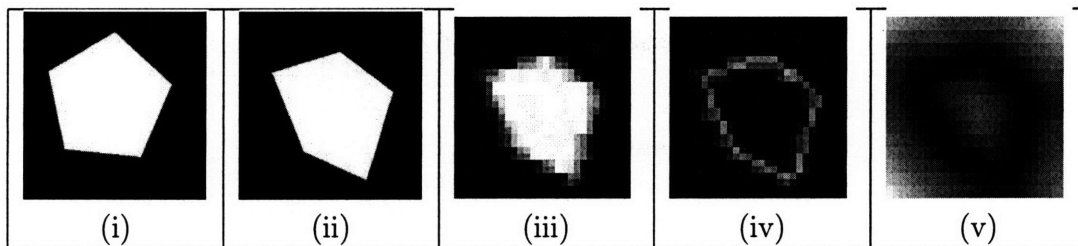
Figure 6-3: The process for creating the toy data. (i) A perfect polygon is created with a given number of vertices. It is enclosed inside a circle of radius 100 pixels and is rotated by a random rotation. (ii) Each vertex is shifted independently by random amount of up to 40 pixels in the x and y directions. (iii) The image is decimated to 28×28 pixels. Two image descriptors are then used: (iv) the edge image, which is computed by using a canny edge detector applied to the decimated image, and (v) the distance transform of the edge image.

6.3.2 The 101 object data set

Here, results are given for the popular 101 classes dataset [37]. The results reported are the average recognition rate obtained in 10 independent trials of 15 random training and 50 different random testing images from each object class; (some classes contained fewer than 65 images in which case all the images not used for training were included in the test set). Using the identical protocol[1], where the errors per class are averaged, Berg reports a performance of 45% on the non-duplicated dataset [7], and Serre reports an average performance of 35% using $10,000$ "global" C_2 features [103]. However, by using the code of [103] combined with a variance normalization step, we get a performance of $36.86\% \pm 1.64\%$ using only $3,000$ of the C_2 features.

To test the performance of the hierarchies on this dataset, we performed two sets of experiments. In the first experiment we used the C_1 features as the low level representation, and their C_2 features as the high level representation. As can be seen in table 6.2 using the right hierarchies significantly improves performance.

The second set of experiments also includes the novel image representations described in chapter 5: Continuity, Circularity and Parallelism (these experiments were performed before the symmetry algorithm was complete). They are used in concert with C_1 as the low-level of the hierarchy, as we believe such representations to exist in

[1]In this borrowed protocol there are 102 classes: background is included as a class and faces and faces-easy are two separate classes.

Method		Recognition rate
C_1	(a)	30.93 ± 1.2
C_2	(a)	36.86 ± 1.6
Both concatenated	(b)	44.18 ± 0.8
Feedforward perception	(c)	43.37 ± 1.0
Reverse hierarchy	(d)	45.81 ± 1.1
Semantic concatenation	(e)	45.16 ± 1.0

Table 6.2: Percentage recognition rate for the 101 dataset where C_1 was used as the low level representation and C_2 was used as the high level one. The results for (b) are given without the matrix normalization step in order to match the results given in Ch. 5. Normalized results are very similar. (e) and (d) perform best followed by (b) and (c).

Method		Recognition rate
C_1 + Cont + Circ + Par	(a)	42.56 ± 1.0
C_2	(a)	36.86 ± 1.6
Both concatenated	(b)	48.26 ± 0.9
Feedforward perception	(c)	50.14 ± 1.2
Reverse hierarchy	(d)	46.97 ± 0.9
Semantic concatenation	(e)	51.18 ± 1.2

Table 6.3: Recognition rate in percent for the 101 dataset, where the low level representation consisted of a combination of C_1, Continuity, Circularity and Parallelism. Semantic concatenation (e) does best followed by (c). (b) and (d) perform worse.

the lower visual areas. As the high-level representation we again use C_2. The results (table 6.3) show that by combining the strategy of semantic concatenation (e) with these image representations one gets the best result on the 101 dataset reported so far.

6.3.3 Street Scene experiments

In this experiment the hierarchy architectures were tested on 3 binary object detection experiments using the car, pedestrian, and bicycle objects from the StreetScenes database [14]. These crop-wise experiments are similar to those performed in previous chapters. Each database consists of labeled positive and negative grayscale examples of the class. The results of these experiments, in terms of the statistics of the ROC

Method		car TP@FP=.01	Eq. err	pedestrian TP@FP=.01	Eq. err	bicycle TP@FP=.01	Eq. err
C_1 Direct.	(a)	82.5±4.9	5.4±1.0	32.9±5.9	19.2±2.4	62.5±8.0	8.6±2.9
C_2 Direct.	(a)	64.1±5.5	7.9±0.9	44.9±5.1	13.5±1.8	49.3±9.7	11.2±2.8
C_1+C_2 Concat.	(b)	85.4±3.6	4.7±1.0	54.2±6.0	12.7±1.5	69.1±7.2	8.2±1.9
FeedFwd. Percept.	(c)	87.1±3.8	4.5±1.1	46.6±10.3	13.9±1.9	69.9±8.3	7.4±2.1
Reverse Hier.	(d)	81.2±5.5	4.8±1.0	60.0±5.3	10.8±1.7	68.7±9.4	7.6±1.8
Semantic Concat.	(e)	86.1±4.3	4.8±1.1	63.7±7.8	10.0±1.9	73.6±8.3	6.4±2.0

Table 6.4: Equal error rate and the true positive rate when the false positive rate is set to 1% for several hierarchical strategies applied to the StreetScenes dataset [15]. The mean and standard-deviation given were computed on 25 independent trails. For each object class, combining the feature modes in a hierarchal manner improves upon any direct classification strategy.

curves produced, are listed in table 6.4. Specifically, the mean and standard deviation of the equal error rate and the true positive rate when the false positive rate is set to 1% are listed. Independent trials consisted of dividing the data into one third for testing, and the rest for training. Boosting, rather than SVM, was used in these experiments.

The first two rows show the result of directly applying a classifier to the raw representations i.e., hierarchy a. In this case, as in all others, the classifier used is gentleBoost, a variant of AdaBoost, which was run to convergence. The concatenation architecture (b), in which the two feature vectors are simply concatenated and then classified, performed better than either feature mode alone for each class. In order to build the other three hierarchy types, it was necessary to build multiple intermediate classifiers, in order to build a perception vector. In these cases, for each necessary group of classifiers, n such intermediate classifiers were constructed by training on random subsets of the examples, a strategy similar to random forests (but without replacement). Each subset contained 20% of the training examples. For the results shown here, n was set arbitrarily to 80, but the results seem to be stable for a wide range of parameter values. Looking at the results, there does not seem to be a clear advantage to either forward (c), or reverse (d), hierarchies, but semantic concatenation (e), seems to consistently perform well.

Method		Eq. err	AUC
Continuity	(a)	9.35 ± 2.1	95.49 ± 1.7
HoG	(a)	7.87 ± 1.5	97.02 ± 0.9
Both concatenated	(b)	6.59 ± 1.6	97.73 ± 0.8
Feedf. perception	(c)	7.33 ± 2.4	97.46 ± 1.1
Reverse hierarchy	(d)	6.66 ± 0.9	98.05 ± 0.5
Semantic conc.	(e)	6.34 ± 1.4	98.19 ± 0.6

Table 6.5: Equal error rate and area under the ROC curve in percent for several hierarchical strategies applied to the pedestrian dataset. The mean and standard-deviation given were computed on 20 independent trails. HoG is better than the continuity representation. Combining them gives best results. Strategies (b), (d), and (e) give best results, while (c) does not seem to do better than HoG.

Another experiment was conducted using the pedestrian data, but this time with the HoG [29] features as the high level representation. The edge continuity feature computed as in Ch. 5 was used as the low level information. In 20 independent trials, the data were split cleanly into 60% training, 40% testing. Here SVM was used. The results shown in table 6.5 suggest that all combination strategies seem better than the baseline, except for the feed-forward perception.

6.3.4 Analysis of the results

The general trends in the experiments seem to be that: (1) classifying a single level of the hierarchy is suboptimal. (2) using perception when combining multiple levels can be beneficial. (3) perception, in most of the experiments, works well when it is applied first to the most discriminative hierarchical level. Finally, semantic concatenation works well in the case where the hierarchy levels are similar in their discriminative power. Below we provide the details of some sanity checks we performed to verify the results.

Are the results statistically significant? Since the same training and testing splits were used for all of the compared algorithms we could verify that the differences are indeed statistically significant. This was done by applying the paired t-test on the set of performance measures returned by the algorithms. In the polygon experiment the semantic concatenation method is significantly better than the others

(p<0.01, corrected for multiple comparisons). In the C_1/C_2 101 object dataset, reverse hierarchy and semantic concatenation are significantly better than the other methods, but not significantly different from each other (p=0.46). In the experiments of table 6.3, semantic concatenation is significantly better than feedforward perception (p=0.005), and the latter is significantly better than all the rest. For the pedestrian experiment, methods (b) (d) and (e) performed significantly better than the rest (p<0.01), but were not significantly different.

Could the source of discrepancy be better normalization? As mentioned above, we normalized the data such that contributions from different sources would be similar. We also performed several experiments examining this normalization. In one experiment we repeated the 101 object experiment using an AdaBoost classifier, which is less sensitive to normalization issues. The results were much worse, in the range of 20% to 30%, but the relative performance of the strategies (a)-(e) remained the same.

In another control experiment we tested concatenation (b) with different scale factors. We used the polygon dataset and scaled the distance map prior to concatenation with a factor that ranged between 10^{-5} to 10^5. None of these experiments resulted in performance higher than 49%.

Maybe adding a classifier on top of a 1-vs-all representation improves results? We tested whether taking a set of features, classifying it with a 1-vs-all classifier to get a vector of perceptions, and then classifying it again improves performance. Intuitively, since the basic 1-vs-all strategy is in the solution space of the two level 1-vs-all, performance may increase. It turns out that it does not. For example, for the C_2 feature set on the 101 object dataset, it reduces performance slightly to 36.10% (from 36.86%).

141

6.4   Conclusions

Our results are too preliminary to tell whether the strategy of reverse-hierarchies is an effective one for object recognition. However, our work, which was inspired by Reverse Hierarchy Theory, has shown that a considerable performance gain can be achieved by employing less common classification strategies for visual hierarchies. One implication is that feedback, even in the limited form studied here, is helpful for object recognition. For the kind of elaborate feedback systems thought to be present in the human visual system, involving attention, object based segmentation, and possibly verification loops, a performance gain in a discriminative framework is much harder to demonstrate. In terms of feature combination strategies which may be attractive to practitioners, the benefit gained from simple concatenation is often significant, but the subsequent increase found in moving to the more complicated combination strategies is often smaller in comparison. This moderate performance increase needs to be weighed against the increased complexity when selecting an appropriate combination strategy.

Chapter 7

Discussion and Conclusions

At the beginning of this thesis we set out to develop a system which could transform a visual input into a useful meaning. In order to narrow the scope and to give the project structure, we decided upon the detection of all examples of several object categories within the scene as a natural and achievable surrogate for the ephemeral term *meaning*. This problem, the accurate detection and localization of objects in natural scenes, has been a focus for many years by many eminent members of the community, but only a fraction of this work has been toward the construction of a *general* solution, *i.e.*, a solution which is applicable to such a wide variety of objects that within a typical scene the majority of the space may be explained by objects detectable within the framework.

This constraint, that for sufficient generality, each object detector should be a simple instantiation of one general framework, restricts space of potential solutions. In particular, it is common practice in *consumer* applications that hand-engineered detectors are built for each object, focusing only on those visual signatures which separate that object from the scene. This solution is undesirable for its complexity and poor scalability; each additional object requires a hand-engineered solution. Our solution, in contrast, does not require re-engineering the internals of the system to detect a new object, but rather to simply replace the set of training images.

By the end of the thesis we had developed one system capable of learning to detect categories as disparate trees and pedestrians, and also demonstrating competitive

error levels which were often better the state of the art in computer vision, more accurate even than systems which had been hand engineered and tuned for one object type on that same object type. This unified approach to scene understanding was made possible through the use of *biologically inspired image representations*, including the Standard Model and the Gestalt extensions, and the judicious use of them through discriminative learning techniques.

Over the course of this project, it was found that in order to build systems of greater discriminative power, or, equivalently, systems which output fewer errors, it is easier to build newer more informative image features, than to design newer more advanced learning techniques. At an early stage we decided to formulate our solution in the discriminative framework, rather than the generative one. We could imagine that, in the generative framework, our solution would have involved designing a distribution which captures the sense of visual object, and tuning this model in different ways depending upon the target object classes. Furthermore, we could imagine imbedding this model within a larger model which captures notions of inter-object influences, and adding a hidden "scene" object which might push these distributions in certain ways. The choice of model is a type of engineering which is in many ways akin to the design of the image features in the discriminative approach. For instance, in the generative framework, we might be wise to postulate that our object is best modeled as some distribution over smaller object "parts" or "fragments," which may at times be occluded or invisible, and have relative geometric constraints. Similarly, in the discriminative framework, it is possible to build detectors of object sub-parts, and use these detections as an image feature for a subsequent classification stage. Our choice to use the discriminative framework stemmed from the higher accuracy rates and lower computational costs which the discriminative approaches have yielded recently, and also to build upon previous results within this framework.

In Ch. 3 it was shown that, compared to many other well respected state of the art image representations, the features derived from Poggio's Standard Model (Ch. 3) were superior for a very wide variety of object detection tasks. What is even more surprising about this finding is that these features were not designed explicitly for

this task, but instead to model part of the human biology. We feel that this may be explained most simply through the process of natural selection, wherein selection pressure pushes the visual system towards representations which make it easier for the organism understand the visual world and thereby thrive. By faithfully copying this representation, we are in effect tapping into the current solution of a computational search process of extraordinary power.

Our implementation of the standard model does not claim to completely model the the human visual search process. It only makes claims about one part, the early feed-forward part, of the ventral stream of the visual cortex. There is a great deal of biology that is intentionally left out, including the higher level executive processes and the and the lateral and feedback connections. This left open avenues for improvement in our *computer* vision system, since there was no way for information from objects to influence each other, or for information from the scene as a whole to influence the detection process. In Chs. 4 − −6 we addressed these by proposing models of interaction between object examples, between objects and the scene itself, and lateral connections within the feature generation structures so as to develop even more powerful image features.

Because these standard model features were built in a way that involved repetitions of certain computational elements and motifs, (a trait which is common in biological systems), it was a small step to recognize that these same motifs could be combined in other ways to build further informative features that might help to further reduce the errors in the object detecting system. In Ch. 5 we described features designed using these same motifs to represent *explicitly* other perceptual stimuli that humans have shown sensitivity to, the Gestalt features. Features which explicitly represent the Gestalt concepts of continuity, circularity, parallelism, and symmetry were designed and implemented, and we demonstrated experimentally that these features were helpful in the object detection task in ways that the standard model was numb to. Further advancements in object detection may possibly be made by looking at these computational motifs more critically, and developing a rigorous search strategy to find those combinations which are most helpful in a general sense.

With the development and embellishment of our computational feature hierarchy, it became more important to investigate the little-understood problem of how best to combine the multiple modes of features from *different levels in the hierarchy* so as to achieve the highest possible levels of accuracy. It was known well before the outset of this project that different objects have different tell-tale characteristics and that, depending upon the object class, a particular feature, *e.g.*, color, may either be either critical or useless. Deciding which features to use, and which to exclude, is a an example of the classic feature selection problem. Our solution space was different from feature selection in that it included different *ways* of inclusion. In an exploration of possible feedback methods, features of one mode could be introduced earlier or later depending on the architecture. The experiments in Ch. 6 describe these experiments. The conclusions were slightly disappointing in that, like choosing which features or learning methods are best for a particular object, the selection of an appropriate hierarchy may be completely problem dependant in a way that requires a training and validation approach. In general, however, it was found that very commonly the simple strategies of either concatenation or semantic concatenation (a.k.a. stacking) were as successful as more complicated approaches, and considerably better than detecting based on one feature mode, or one level of the feature hierarchy.

While the additional features and compositional strategies were very helpful in the detection of individual objects, it was still felt that detection could be further improved if there was some way for the presumptive object detections to influence one another so as to remove spurious object detections which did not fit well with the regular structure of the street-scenes. In Ch. 4 we designed and measured the effectiveness of a context system which allowed the object detectors to do exactly this, all within a discriminative framework. In the development of this system we demonstrated the use of both general and relative position information. We also performed a series of experiments exploring the way that contextual information is best extracted from the image. In particular, we showed that strong contextual information, *i.e.*, information not due to the object's appearance but which can nevertheless be used reliably to influence detection confidences, can be extracted at a low level or

a high level, and there is little difference in the utility of the two. This important finding suggests that, in the building of contextual information for one object, there is little need to detect other related objects in the scene, rather, a strong model of object context can be developed by sampling color, brightness, and texture features using a larger receptive field. Another surprising finding from our experiments in context involved the relative utility of context and appearance information. We showed that, at least for these types of scenes, a reliable detector of object appearance is not helped much by a reliable detector of object context. That is to say that once an candidate has been found that appears enough like our target that the appearance detector will pass threshold, the candidate is very rarely so far out of context that a combination strategy will reject it. This does not make any claims about situations when the appearance information is weak, such as when the object is blocked out, blurred out, of low resolution, or otherwise invisible. In these situations the context information remains very useful to the detection process.

Through the course of the development of this system, we have strived to compare fairly to the state of the art in computer vision, and to provide reproducible measures of performance. Recently it has become the trend in computer vision that methods and procedures are judged based upon their performance on common public tests. By making our database public we hope to promote healthy competition and the sharing of successful methods in *scene-understanding*. In Ch. 2 we described the choices we made during the development of the StreetScenes database and the associated performance measures. Briefly, there was a need for a database which involved the detection of multiple varied object types, all within the same scenes, so as to pave the way for the development of more general object detectors, as well as systems for contextual modulation. By building a system which is capable of detecting a wide variety of objects in these scenes, building reproducible measures of performance of this system, and then improving upon this system through the addition of context, gestalt features, and feedback, the the work of this thesis has moved us closer to synthetic vision at human accuracy, and better understanding of vision in general.

Chapter 8

Future Directions for StreetScenes

In the course of an academic endeavor, there are inevitably more questions than answers, and more opportunities than time to follow through on them. In this chapter several potentially fruitful pathways for future work are described, along with a few thoughts about what might be accomplished by working in these areas.

8.1   Video Street Scenes

When a child learns to understand the world around it, it has several advantages that the system we have described does without, such as stereo information, and the ability to interact with the visual world. Perhaps the most important among these advantages, however, is the fact that children learn from a constant stream of visual input, rather than a set of dissociated still images. In effect, children learn from one constant video stream. At first it may seem that the additional complexity of working with the additional time dimension can add nothing but difficulty to the scene understanding task. It may be the case, however, that the additional richness of the video stimulus might provide a phase shift which could enable previously impossible capabilities. For instance, in video, there is the opportunity to learn the segmentation of foreground objects from background almost automatically, due to parallax motion. There is no similar cue in still images which would allow us to separate objects

In fact, there is evidence that the visual system makes heavy use of these sorts

motion patterns in the bootstrapping of the human visual system. Psychophysical studies of object constancy in young infants have shown that they have more trouble recognizing still stimuli than moving ones, and only later in development do infants become as adept at forming object candidates in still stimuli as in moving ones. This order of visual understanding is supported in studies of late-stage reversible blindness, wherein subjects of life-long blindness are healed, and their ability to see is closely monitored [57, 85]. It may be that the most useful representations for vision may be learned most successfully by learning from video, rather than still scenes, even if, in the end, the system will only be used to analyze still stimuli.

In addition to the potential for greater accuracy or complexity of the output systems, there are other reasons to move to video. For instance, working in video would allow for the recognition of actions, interactions, and behaviors, as well as still objects. This richer problem could potentially lead to useful technologies in the fields of surveillance, security, health-care, and many others.

One problem in moving to recognition in video would be in the development of an appropriate database for training and testing. The development of the Street Scenes database, which contains fewer than 10,000 images, cost hundreds of man-hours of labor. Even one hour of video at 15 frames per second would require the labeling of 15,000 frames. In order to build a labeled database large and diverse enough to perform meaningful examples, we would expect that some automatic method would be employed. It may be possible, for instance, to build a video database of a street-scenes like environment with completely perfect labeling using computer graphics. Modern computer graphics, of the level used in high-end video games, is probably realistic enough to capture the types of invariance necessary to build a robust system that could function accurately in a real world setting.

8.2   Reconciling the texture and shape pathways

In the development of any system, simple solutions are preferable to complex ones. The most striking signature of hand-engineering in the current implementation of the

scene-understanding mechanism may be that two separate modes of processing are used to detect the two classes of objects, texture-based objects, such as the buildings and trees, and shape-based objects, like the cars and pedestrians. While this isn't so much of a problem in terms of feasibility, a simpler one-path solution would be desirable.

One possible solution would be to build an object detector of both the shape-based and texture-based type for each object class. For instance, one might consider building not only a shape-based car detector, using windowing to detect cars across position and scale, but also a texture-based car detector, which would presumably classify each pixel as either car or non-car. How to reconcile the outputs of the two detectors is still an unknown, but such a design might help in those situations where large crowds of cars or pedestrians makes it difficult to detect any single one.

8.3   Computational cost

Currently, it requires several minutes of processing to convert a full size StreetScenes image into the standard model format, and more time afterwards to scan this representation for each object class. In any practical use of this sort of system, computational cost is likely to be a major concern. Since the system was designed in MATLAB without consideration for running time, there are many avenues for possible improvement. Firstly, it is very likely that the high cost operations, such as normalized cross correlation and linear filtering can be implemented faster than their current form. Secondly, it may be possible to use approximations such as PCA to drastically reduce the computational cost of the C_2 layer. Finally, since a great deal of the computation consists of parallel arrays of simple operations, the entire process may be farmed out to some heavily parallel hardware, such as a modern GPU, for ultra fast computation. It is likely that a functional form of the feature-generation and object-detection architecture could be run at near frame rate.

Appendix A

StreetScenes Detections and Label Examples

In this section we will illustrate several examples of the StreetScenes Detections database, including illustrations of the labels, the output of our complete scene understanding system, and some comments for each example. Note that these figures are best viewed in color, and it may be very difficult to make sense of the the StreetScenes detections in grayscale. Each example will include the following data:

Original image The original image is a reproduction of the actual image from the database. The original images are stored in JPEG format and are 960 × 1280 pixels in resolution.

StreetScenes Detections This figure illustrates the output of our current scene-understanding system. The output of the texture-based object detectors is indicated by the pixel tint overlay, as in chapter 3. Brown tint indicates building, green for trees, blue for skies, and the grey/black tint indicates road. The bounding boxes show the detection of the shape based objects. Red boxes circumscribe car detections, and light blue for pedestrians. These results do not include the Gestalt features, which have been shown to cut error rates significantly, but have not been tested in a windowing framework. It is likely that some, but not all, of the errors could be remedied by the addition of the Gestalt features.

Label image For each of the 9 labeled object types we will include an illustration

of the hand drawn polygons associated with this example. A blank image indicates that there were no labeled examples of this object type in this example. All SS objects are labeled with closed polygons, but since these polygons sometimes exit the image, the illustrations produced here sometimes appear as open figures.
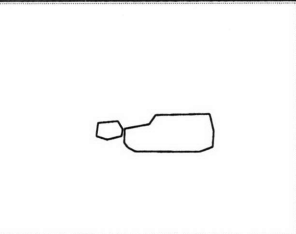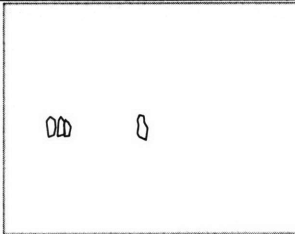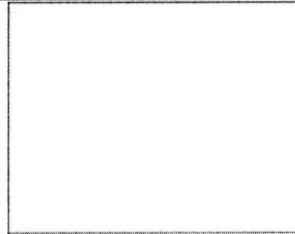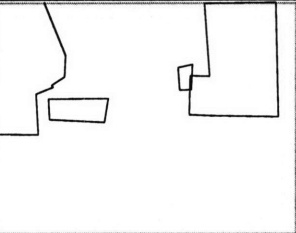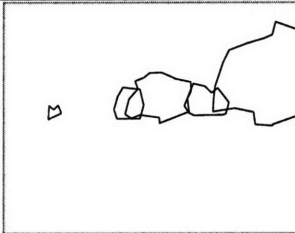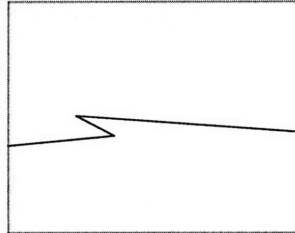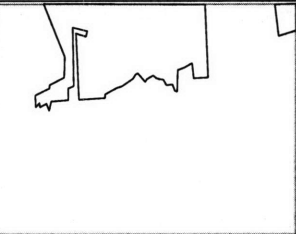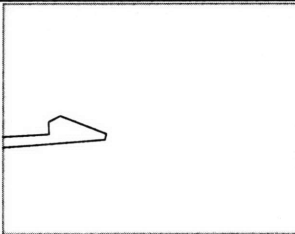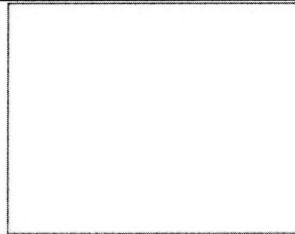
Original image	StreetScenes Detections

Ground Truth Labels		
Car	Pedestrian	Bicycle
Building	Tree	Road
Sky	Sidewalk	Store

Figure A-1: **StreetScenes Index 1**: This image is notable only in that it is the first of the 3,547 scene examples in the StreetScenes database. The system seems to do well in detecting the pedestrians, and also detects the smaller car in the background, but it missses the large SUV.
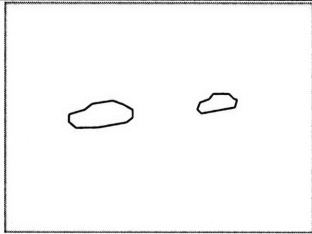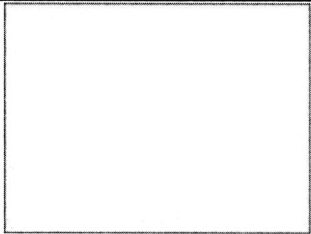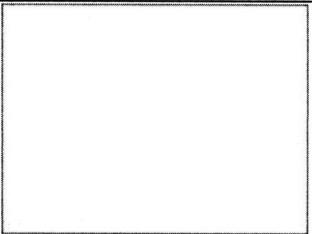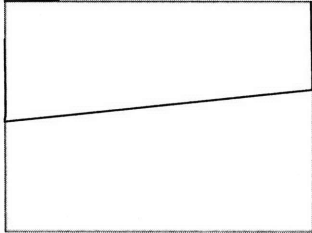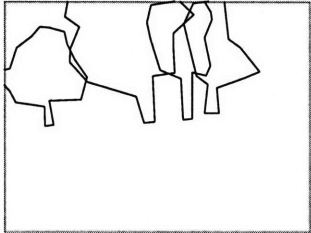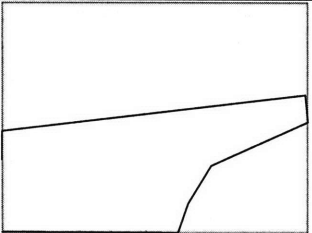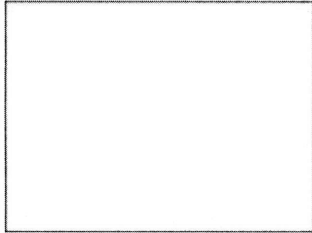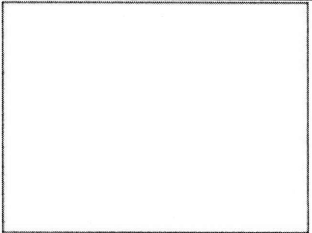
Original image	StreetScenes Detections

Ground Truth Labels		
Car	Pedestrian	Bicycle

Building	Tree	Road

Sky	Sidewalk	Store

Figure A-2: **StreetScenes Index 3**: The white street sign is mistakenly recognized as sky. The front car is detected, but the back car is not.
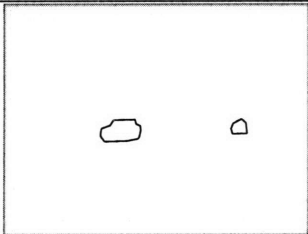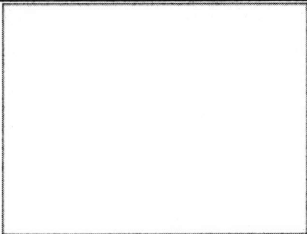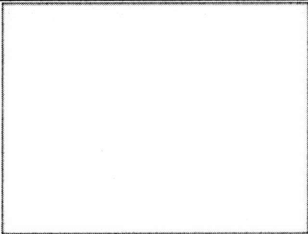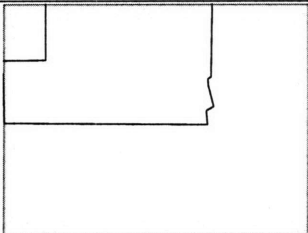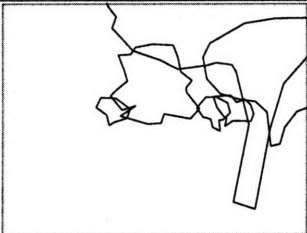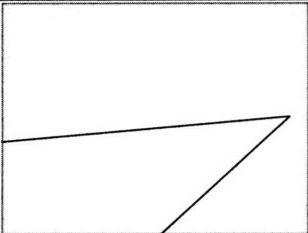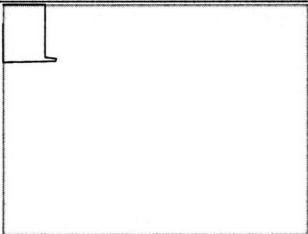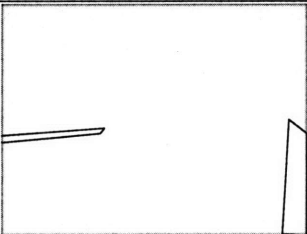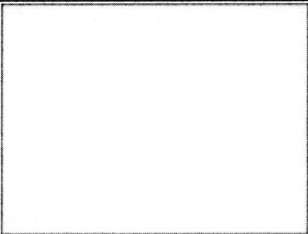
Figure A-3: **StreetScenes Index 5**: The car is detected correctly. Note how the trees are detected correctly, even including the trunk, but the green box to the left of the car is correctly identified as building.
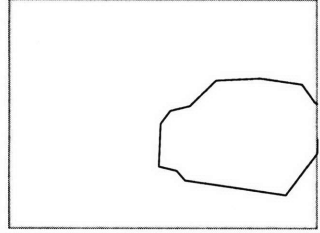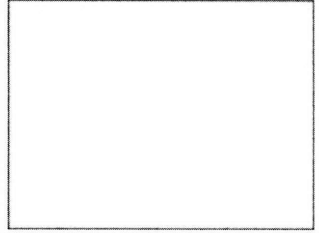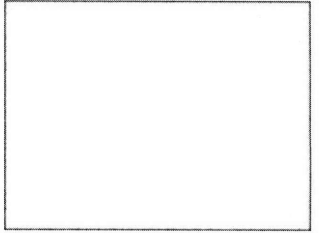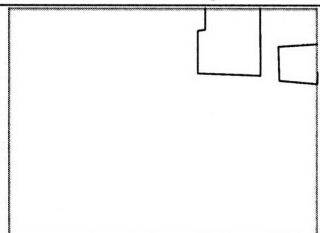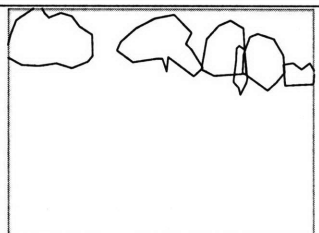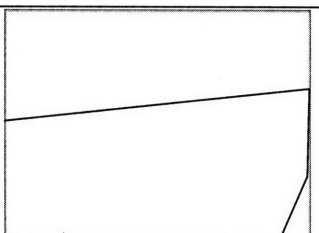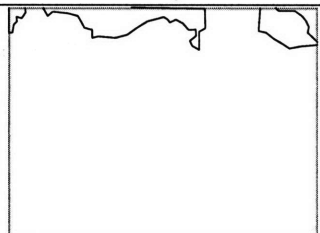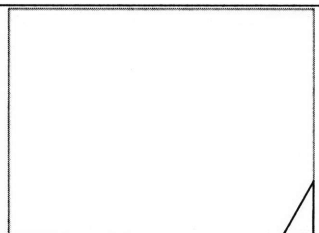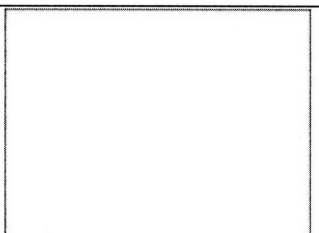
Original image	StreetScenes Detections

Ground Truth Labels		
Car	Pedestrian	Bicycle

Building	Tree	Road

Sky	Sidewalk	Store

Figure A-4: **StreetScenes Index 24**: The large car is beneath the threshold detection level, and remains undetected. It might be far enough beyond the border of the image that the windowing process was unable to detect it. Note also the confusion the system has in labeling the texture objects over the car. It may be a good idea to develop a car *texture* detector in a future version.
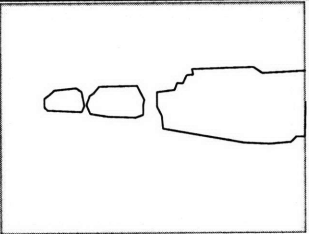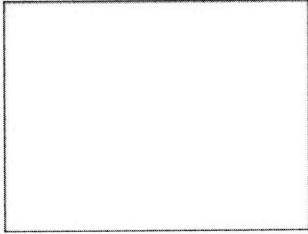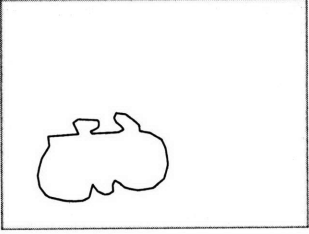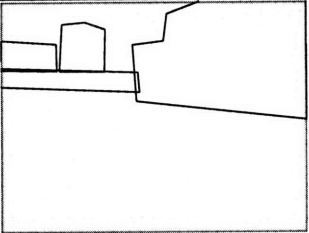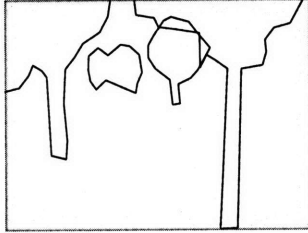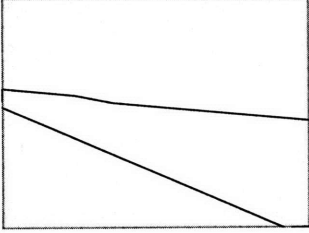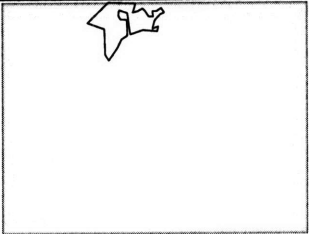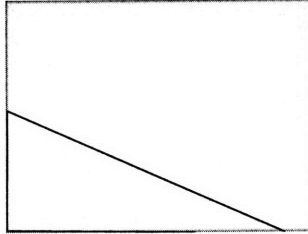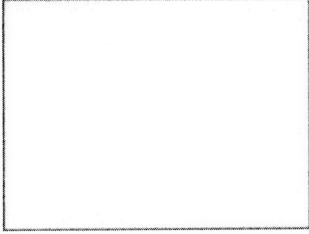
Figure A-5: **StreetScenes Index 62**: The front wheel of the truck is mistaken as a full car detection. Note how the street detector does not mistake the brick sidewalk for building.
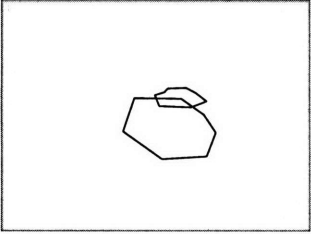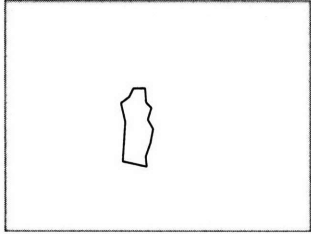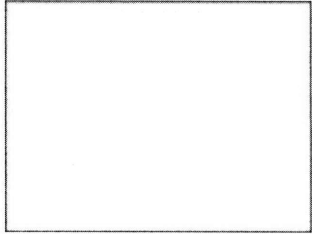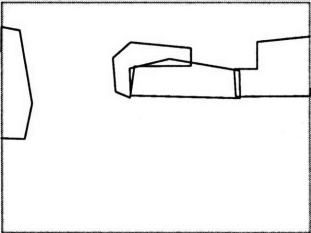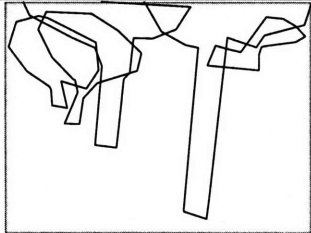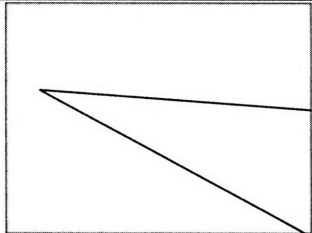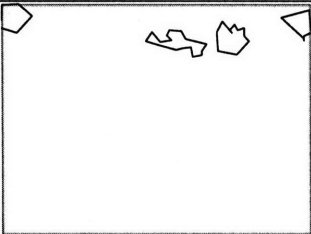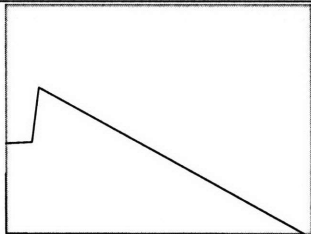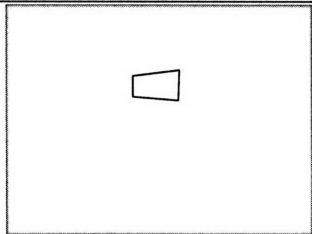
Original image	StreetScenes Detections

Ground Truth Labels		
Car	Pedestrian	Bicycle
Building	Tree	Road
Sky	Sidewalk	Store

Figure A-6: **StreetScenes Index 240**: The large pedestrian is detected correctly, but no vehicles are detected Most of the vehicles are heavily occluded, so this is a particularly difficult example.
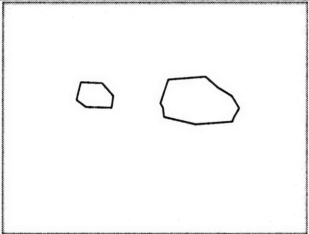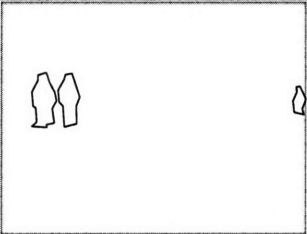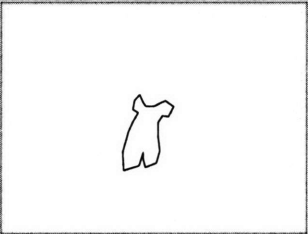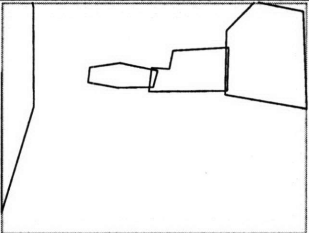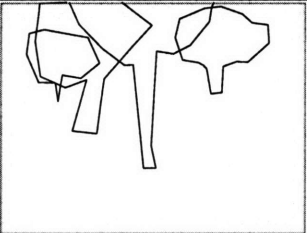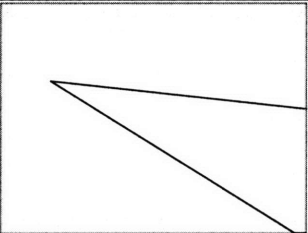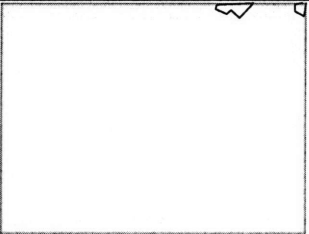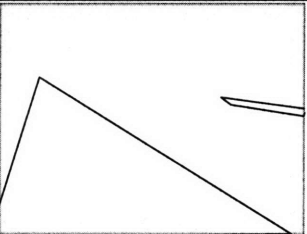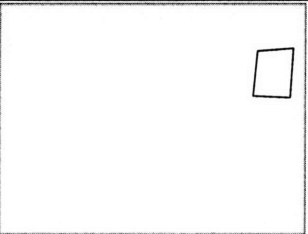
Original image	StreetScenes Detections

Ground Truth Labels

Car	Pedestrian	Bicycle
Building	Tree	Road
Sky	Sidewalk	Store

Figure A-7: **StreetScenes Index 243**: In this example both pedestrians are detected accurately, and there are few mistakes in the texture recognition. The larger car is a false negative.
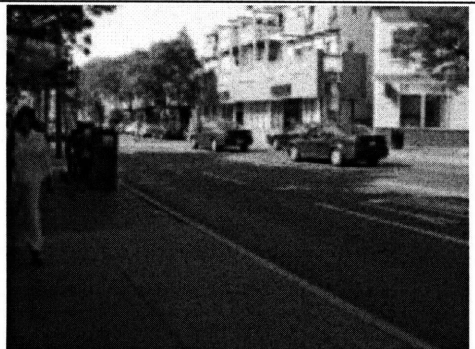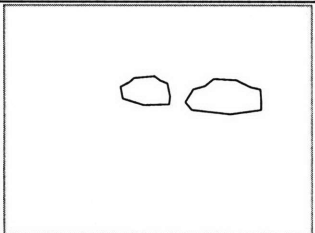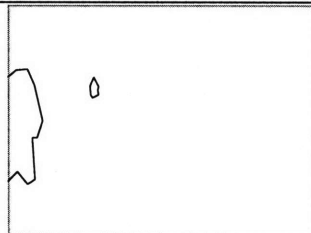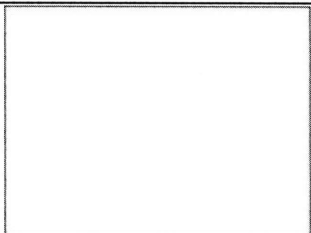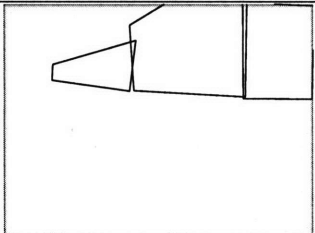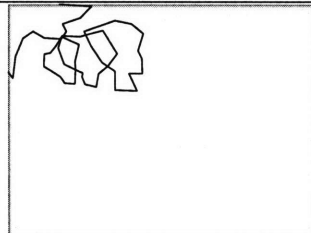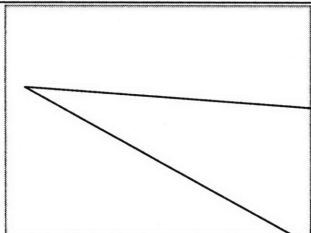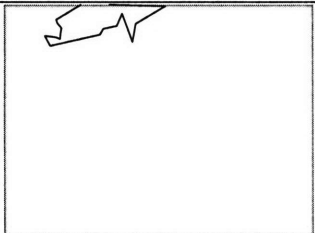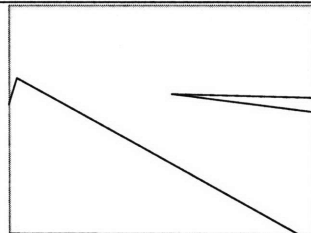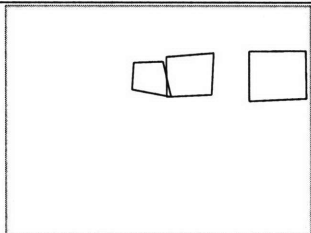
Original image	StreetScenes Detections

Ground Truth Labels

Car	Pedestrian	Bicycle

Building	Tree	Road

Sky	Sidewalk	Store

Figure A-8: **StreetScenes Index 247:** This image includes two false positive pedestrian detections, the left most detection is actually detecting a news-box. There is one true-positive car detection, but the larger car is not detected accurately. The car behind the largest one might be a true-positive if it were labeled. It is likely unlabeled due to heavy occlusion.
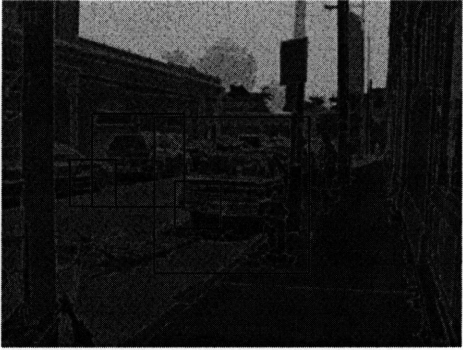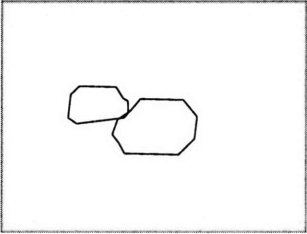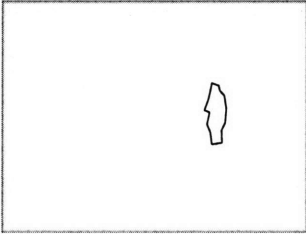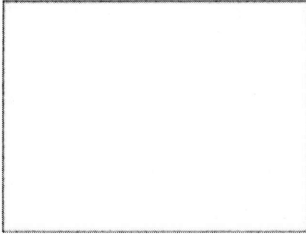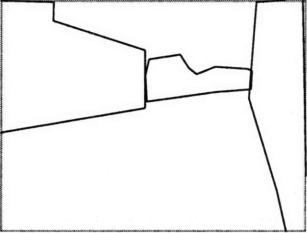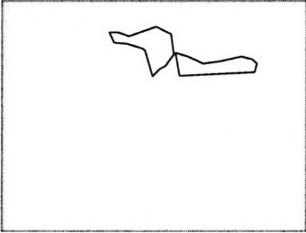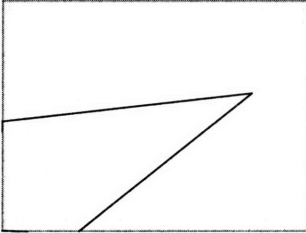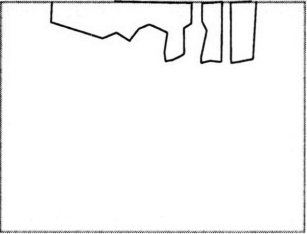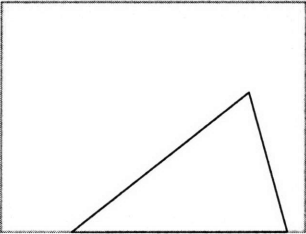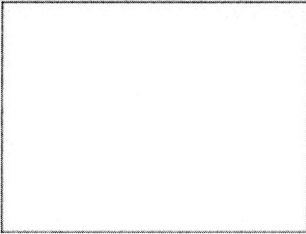
Original image	StreetScenes Detections

Ground Truth Labels

Car	Pedestrian	Bicycle

Building	Tree	Road

Sky	Sidewalk	Store

Figure A-9: **StreetScenes Index 248**: Two of the lesser occluded cars are correctly detected, but there are several smaller false positives. These types of false positives are common to this system, since negatives of this sort were excluded from the training set.

163

Appendix B

How to Work with StreetScenes MATLAB Code

In this section we will discuss the interfaces to the major pieces of code designed to work with the StreetScenes database. These include all programs to extract data, to train and test classifiers, and to measure system performance. Also included with the StreetScenes database are a host of additional helper functions, dealing with image transforms, translation to and from the LabelMe format, and statistical manipulation of the experimental data, such as the generation of ROC curves, among other necessary funcitons. These functions are not described here, internal documentation should be sufficient for understanding their function. Note that the standard model code is not part of the StreetScenes database, and can be downloaded separately at http://cbcl.mit.edu.

B.1 General Purpose Code

SSDB_InitializeDirectories:

This function saves two matlab files to the path string stored in DBRoot. These files are referenced in other interface files for information such as the locations of the images and labels within the database. This function must be run at least once with the correct pathname of the database in order for the remainder of the functions to work properly

B.2 Crop-wise Object Detection Code

```
SSDB_Experiment_BuildCropDataset
```

This function searches all or part of the StreetScenes database (SSDB) for examples of one target object, and extracts positive and negative examples of this object to disk. This extraction can be used subsequently for crop-wise object detection experiments. Each extraction will be converted to grayscale and resized to a common size before storage to disk. The structure options can optionally change much of the default operation of this function. The target object is controlled by the string contained in the field "ObjectName" of options. Similarly, it is possible to control which SSDB example scenes are to be extracted from, what minimum object size to extract, the resolution the extractions will be recorded at, and the number of negative extractions to take per positive example. Other, less consequential parameters can also be changed from their default behavior via the options object. Negative examples are always extracted using a distribution of sizes which matches the distribution of the positive examples. Positive crop locations are selected by looking up the object bounding polygon, computing the minimum bounding rectangle, extending the rectangle so that the aspect ratio will be $1:1$, and then further padding this square with an additional $\frac{1}{3}$ of its size, so that there is some room near the border for edge operators to work. Bounding box locations are only extracted if they are completely within the scene. Negative locations are only extracted if they overlap locations of true positives by less than 10% of the area of the negative box. An undesirable side-effect of this measure of separation between positive and negative examples is that no negatives are extracted from very small boxes within a large object.

```
SSDB_Experiment_CropwiseObjRec
```

This function is an example of how to use the crop-wise object methodology to generate the ROC curves. It takes as input cell arrays indicating the location(s) of matrices saved in the format output by SSDB_Experiment_BuildCropDataset, and trains and tests a classifier based on the contained data. By supplying matrix locations, it is possible to train classifiers based on the concatenation of multiple features at once. The function decides which examples to use for training and which for testing based on a random split, which can be

made deterministic by supplying the random seed in the options structure. It is also possible to control the statistical learning machine and data normalization procedures through this structure. The output of this method is an ROC curve, associated statistics, and the learned model. In order to build the statistics contained within this thesis, this function was run multiple times with preset random seeds. In this way it was possible to compare different features or different classification algorithms on the same example train/test splits using a paired t-test.

B.3 Pixel-wise Object Detection Code

SSDB_Experiment_PixelwiseObjRec

This function returns a structure of testing and training pixel locations from within the SSDB. This structure, marking locations of positive object presence, can be passed to the function SSDB_RecordPixelwiseJets to record actual samples at these points. The parameters controlling this function include which SSDB images are to be included, which objects to sample from, and how far from the object border to exclude. In the pixel-wise object detection paradigm, unlike the crop-wise, then negative examples are selected based on positive examples of other objects. Thus, it doesn't make much sense to extract locations of only one one type of object.

SSDB_RecordPixelwiseJets

This function takes the pixel location structure output from SSDB_Experiment_PixelwiseObjRec, and a parameter describing the location of the SSDB converted into some useful format, such as blobworld or C_2, and returns this representation at the selected pixels. The function writes these jet matrices to disk so that the user can train and test classifiers on them. It is possible for this function to record the local area of the pixel, a small patch, instead of just the jet exactly at the pixel.

SSDB_Evaluate_PixelwiseObjRec

This function measures the performance of classifiers trained and tested on the data

output from SSDB_RecordPixelwiseJets. The output consists of ROC curves and related statistics for each binary texture-based object detection problem.

B.4 Box-wise Object Detection Code

CollectBaselineDetections

This function returns a list of ground truth locations for some set of objects within the SSDB. The output is a struct array in which each element describes one object example, and is labeled with the object class, the origin image, and the bounding box surrounding the object. This struct array becomes an input for the SSDB_Experiment_BBoxwiseObjRec function so that these ground-truth locations can be compared to the machine-detections. The ground truth locations can be controlled by choosing which SSDB images to allow, how large the minimum detectable size is, and whether or not to mandate that all detections should have an aspect ratio of exactly 1. If elongated detection boxes are allowed, then the problem becomes slightly more difficult in that each detection must not only localize the object, but also perform a crude segmentation so as to chose a bounding box of appropriate shape.

WindowedObjectDetection

This function extracts windows of an image at many locations and scales and passes each one to a classifier. The output is a set of detection strengths over the scale space. Default parameters involve detecting a 128 × 128 pixel box at one pixel offsets in space, and at a geometric set of scales from the original size down to a size just larger than the size of the window. The default spacing is a factor of $2^{.25}$. Practical solutions can use a much coarser grain to process more quickly without losing too much accuracy.

LocalNeighborhoodSuppression

The output of WindowedObjectDetection function is a three dimensional structure describing the confidence of object detection over space and scale, but what is necessary for the box-wise detection measure is a set of candidate locations and the confidences at those

locations. Since classifiers tend to be tolerant to some space and scale transformations is necessary to perform local neighborhood suppression so that each object is detected only once. This local neighborhood suppression algorithm works by iteratively recording the global maximum of the space, along with the corresponding bounding box, and then decimating the local region by multiplying the local region by 1 minus a small gaussian centered at the detection location. In this way, near locations are reduced in strength, but distant regions are unaffected. The size of the gaussian is a parameter of the algorithm, but it is further modified by the detected scale so that large objects remove large regions of the space, but small objects only decimate the local region.

| SSDB_Experiment_BBoxwiseObjRec |

The box-wise object detection paradigm involves attempting to build a pairwise correspondence between the detected locations and ground-truth baseline locations. A detection is determined to be a true positive if it has the same class label as some ground-truth example, overlaps it by $\tau\%$ of the union of their areas, and the ground-truth example hasn't been detected before, where τ is a controllable parameter of the algorithm and defaults to 50%. This pairwise matching can become arbitrarily complicated, especially in the case when τ is very large and there are many overlapping detections. Our matching algorithm is a fast approximation to the optimal solution, and works very well in the types of detection scenarios found in the SSDB.

Bibliography

[1] Shivani Agarwal and Dan Roth. Learning a sparse representation for object detection. In *Proceedings of the 7th European Conference on Computer Vision*, volume 4, pages 113–130, 2002.

[2] F. Attneave. Some informational aspects of visual perception. *Psych. Rev.*, 61:183–193, 1954.

[3] M. Bar, E. Aminoff, J. Boshyan, M. Fenske, N. Gronauo, and K. Kassam. The contribution of context to visual object recognition. *Journal of Vision*, 5(8), 2005.

[4] Evgeniy Bart and Shimon Ullman. Cross-generalization: Learning novel classes from a single example by feature replacement. In *CVPR*, 2005.

[5] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 2002.

[6] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR*, 2005.

[7] A.C. Berg and J. Malik. Geometric blur for template matching. In *Computer Vision and Pattern Recognition (CVPR)*, 2001.

[8] N. H. Bergboer, E. O. Postma, and H. J. van den Herik. Context-enhanced object detection in natural images. *In Proceedings of the Belgian-Netherlands AI Conference (BNAIC)*, 2003.

[9] I. Biederman, R. C. Teitelbaum, and R. J. Mezzanotte. Scene perception: A failure to find a benefit from prior expectancy or familiarity. 9:411–429, 1983.

[10] S. M. Bileschi. Streetscenes database ai memo : Under construction. 2006.

[11] S. M. Bileschi. The streetscenes scene undersstanding database and performance measures. *To Be Published as AI MEMO*, 2006.

[12] S. M. Bileschi and B. Heisele. Advances in component-based face detection. In *Proceedings of Pattern Recognition with Support Vector Machines, First International Workshop, SVM 2002*, pages 135–143, Niagara Falls, 2002.

[13] S. M. Bileschi and L. Wolf. A unified system for object detection, texture recognition, and context analysis based on the standard model feature set. In *British Machine Vision Conference (BMVC)*, 2005.

[14] S. M. Bileschi and L. Wolf. Image representations beyond histograms of orientations: Nonlinear gestalt descriptors, 2006. Submitted.

[15] S. M. Bileschi and L. Wolf. Image representations beyond histograms of orientations: Nonlinear gestalt descriptors. In *TO BE PUBLISHED AS AN AI MEMO*, 2006.

[16] V. Blanz, S. Romdhani, and T. Vetter. Face identification across different poses and illuminations with a 3d morphable model. In *Proc. of the 5th Int. Conference on Automatic Face and Gesture Recognition (FG)*, pages 202–207, 2002.

[17] Eran Borenstein and Shimon Ullman. Class-specific, top-down segmentation. In *ECCV*, pages 109–122, 2002.

[18] A.C. Bovik, M. Clark, and W.S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:55–73, 1990.

[19] L. Breiman. Random forests. *J. Machine Learning*, 2001.

[20] Jean Bullier. Integrated model of visual processing. *Brain Research Reviews*, 36:96–107, 2001.

[21] J. Canny. A computational approach to edge detection. *PAMI*, 8, 1986.

[22] Freitas N. Carbonetto, P. and K. Barnard. A statistical model for general contextural object recognition. *ECCV*, 2004.

[23] C. Carson, M. Thomas, S. Belongie, J. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. 1999.

[24] Chih-Chung Chang and Chih-Jen Lin. *(LIBSVM): alibrary for support vector machines*, 2001.

[25] C. M. Christoudias, B. Georgescu, and P. Meer. Synergism in low level vision. volume IV, pages 150–155, August 2002.

[26] D. Cox, E. Meyers, and P. Sinha. Contextually evoked object-specific responses in human visual cortex. *Science*, 304:115.

[27] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. *European Conference on Computer Vision*, 8, 2004.

[28] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[29] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[30] J.G. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *J. Optical Society of America A*, 2:1160–1169, 1985.

[31] S.J. Dickinson, A.P. Pentland, and A.Rosenfeld. From volumes to views: an approach to 3-d object recognition. *CVGIP: Image Underst.*, 55(2):130–154, 1992.

[32] G. Dorko and C. Schmid. Selection of scale invariant neighborhoods for object class recognition. In *International Conference on Computer Vision*, 2003.

[33] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. *Proceedings of SIGGRAPH 2001*, pages 341–346, August 2001.

[34] Boris Epshtein and Shimon Ullman. Feature hierarchies for object classification. In *ICCV*, pages 220–227, 2005.

[35] Manfred Fahle and Tomaso Poggio. *Perceptual Learning*. MIT Press, Cambridge, USA, 2002.

[36] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR, Workshop on Generative-Model Based Vision*, 2004.

[37] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, volume 2, pages 264–271, 2003.

[38] Perona P. Fink, M. Mutual boosting for contextual inference. *NIPS*, 17, 2003.

[39] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *Proc. of the Intl. Workshop on Automatic Face- and Gesture-Recognition*. Zurich, 1995.

[40] William T. Freeman, Egon C. Pasztor, and Owen T. Carmichael. Learning low-level vision. *Intl. Journal of Computer Vision*, 40:25–47, 2000.

[41] D. Gabor. Theory of communication. *Journal of the IEE*, 93:429–457, 1946.

[42] W. S. Geisler, J. S. Perry, B. J. Super, and D. P. Gallogly. Edge co-occurence in natural images predicts contour grouping performance. *Vision Research*, 41:711–724, 2001.

[43] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, pages 1458–1465, 2005.

[44] W. Eric L. Grimson and Tomás Lozano-Pérez. Model-Based Recognition and Localization from Sparse Range or Tactile Data. *International Journal of Robotics Research*, 3(3):3–35, Fall 1984.

[45] Riseman E.M. Hanson, A.R. Visions: A computer system for interpreting scenes. *Computer Vision Systems*.

[46] R. M. Haralick. Decision making in context. *PAMI*, 1983.

[47] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision II*, volume 1. Addison-Wesley, first edition, 6 June 2002. This is a full INBOOK entry.

174

[48] T. Hastie, R. Tibshirania, and J. Friedman. *The Elements of Statistical Learning*, chapter 8. Springer, 2001.

[49] J. Hawkins. *On Intelligence*. Times Books, 2004.

[50] B. Heisele, P. Ho, and T. Poggio. Face recognition with support vector machines: Global versus component-based approach. In *Proc. 8th International Conference on Computer Vision*, volume 2, pages 688–694, Vancouver, 2001.

[51] B. Heisele, T. Serre, M. Pontil, and T. Poggio. Component-based face detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 657–662, Kauai, 2001.

[52] S. Hochstein and M. Ahissar. View from the top: Hierarchies and reverse hierarchies in the visual system. *J. Neuron*, 36(5):791–804, 2002.

[53] Pashler H Huang, L. and J Junge. Are there capacity limitations in symmetry perception? *Psychonomic Bulletin & Review*, in press.

[54] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[55] T. Hastie J. Friedman and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Ann. Statistics*, 2000.

[56] Scott P. Johnson and Richard N. Aslin. Perception of object unity in 2-month-old infants. *Developmental Psychology*, 31(5):739–745, 1995.

[57] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision.*, 1987.

[58] Krzysztof Krawiec and Bir Bhanu. Visual learning by coevolutionary feature synthesis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(3):409–425, 2005.

[59] H. Kruppa and B. Schiele. Using context to improve face detection. *British machine Vision Conference (BMCV)*, 2003.

[60] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. *ICCV*, 2003.

[61] Yoshinori Kuno, Katsushi Ikeuchi, and Takeo Kanade. Model-based vision by cooperative processing of evidence and hypotheses using configuration spaces. In *Proceedings of SPIE Vol. 938 Digital and Optical Shape Representation and Pattern Recognition*, pages 444 – 453, April 1988.

[62] S. Lazebnik, C. Schmid, and Jean Ponce. A maximum entropy framework for part-based texture and object recognition. In *ICCV*, 2005.

[63] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using local affine regions. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 27, pages 1265–1278, August 2005.

[64] Y. LeCun, F.-J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, 2004.

[65] Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR (2)*, pages 97–104, 2004.

[66] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *SLCP '04 Workshop on Statistical Learning in Computer Vision*, 2004.

[67] B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *BMVC*, 2003.

[68] T. K. Leung, M. C. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. In *Proc. International Conference on Computer Vision*, pages 637–644, Cambridge, MA, 1995.

[69] Fei-Fei Li, R. Fergus, and P. Perona. A bayesian approach to unsupervised one-shot learning of object categories. In *Ninth IEEE International Conference on Computer Vision (ICCV)*, volume 2, page 1134, 2003.

176

[70] Fei-Fei Li, R. Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE CVPR Workshop of Generative Model Based Vision (WGMBV)*, 2004.

[71] Fei-Fei Li and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 524–531, Washington, DC, USA, 2005. IEEE Computer Society.

[72] T. Lindeberg. Scale-space theory: A framework for handling image structures at multiple scales. In *Proc. CERN School of Computing, Egmond aan Zee, The Netherlands.*, 1996.

[73] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2003.

[74] David G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.

[75] G. Marola. Using symmetry for detecting and locating objects in a picture. *Comput. Vis. Graph. Image Process.*, 1989.

[76] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of 3d shapes. In *Proceedings of the Royal Society of London. Series B,*, volume 200, 1978.

[77] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001.

[78] Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[79] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. In *PAMI*, volume 23, pages 349–361, 2001.

[80] Kevin Murphy, Antonio Torralba, and William T. Freeman. Graphical model for recognizing scenes and objects. In Sebastian Thrun, Lawrence Saul, and Bern-

hard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[81] S. Nene, S. Nayar, and H. Murase. Columbia object image library: Coil, 1996.

[82] K. Okada, J. Steffens, T. Maurer, H. Hong, E. Elagin, H. Neven, , and C. von der Malsburg. The bochum/usc face recognition system and how it fared in the feret phase iii test. In *Face Recognition: From Theory to Applications*, pages 186–205, 1998.

[83] B.A. Olshausen and D.J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[84] Andalman A. Ostrovsky, Y. and P. Sinha. Vision following extended congenital blindness. Under Review, 2006.

[85] E. Osuna. *Support Vector Machines: Training and Applications*. PhD thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, 1998.

[86] E. Osuna, R. Freund, and F. Girosi. Training support vector machines:an application to face detection. In *CVPR*, 1997.

[87] D.I. Perrett and M.W. Oram. Neurophysiology of shape processing. *Imaging Vis. Comp.*, 11:317–333, 1993.

[88] Tomaso Poggio and Kah Kay Sung. Finding human faces with a gaussian mixture distribution-based face model. In *ACCV*, pages 437–446, 1995.

[89] Molly C. Potter. Short-term conceptual memory for pictures. *Journal of Experimental Psychology*, 2:509–522, 1976.

[90] R. Shanmugan R. Haralick and I. Dinstein. Tetural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6):610–621, 1973.

[91] L. W. Renninger and J. Malik. When is scene recognition just texture recognition? In *Vision Research*, volume 44, pages 2301–2311.

[92] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.

[93] R. Rifkin, G. Yeo, and T. Poggio. Regularized least squares classification. *Advances in Learning Theory: Methods, Model and Applications*, 190, 2003.

[94] T. D. Rikert, M. J. Jones, and P. Viola. A cluster-based statistical model for object detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1046–1053, Fort Collins, 1999.

[95] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *MIT AI Lab Memo AIM-2005-025*, September 2005.

[96] E. Saber and A. Tekalp. Face detection and facial feature extraction using color, shape and symmetry based cost functions. In *Proc. International Conference on Pattern Recognition*, volume 1, pages 654–658, Vienna, 1996.

[97] T.D. Sanger. Stereo disparity computation using gabor filters. *Biological Cybernetics*, 59:405–418, 1988.

[98] Georg Schneider, Heiko Wersing, Bernhard Sendhoff, and Edgar Körner. Evolutionary optimization of a hierarchical object recognition model. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 35(3):426–437, 2005.

[99] H. Schneiderman. *A Statistical Approach to 3D Object Detection Applied to Faces and Cars*. PhD dissertation, Carnegie Mellon University, Department of Computer Science, May 2000.

[100] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 45–51, Santa Barbara, 1998.

[101] T. Serre, L. Wolf, S. M. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *PAMI*, 2006. 'To appear'.

[102] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, 2005.

[103] Amnon Shashua and Shimon Ullman. Grouping contours by iterated pairing network. In *NIPS-3: Proceedings of the 1990 conference on Advances in neural information pro-*

cessing systems 3, pages 335–341, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[104] Dinggang Shen, Horace H.S. Ip, Kent K.T. Cheung, and Eam Khwang Teoh. Symmetry detection by generalized complex (gc) moments: A close-form solution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):466–476, 1999.

[105] Terence Sim, Simon Baker, and Maan Bsat. The cmu pose, illumination, and expression (pie) database of human faces. Technical Report CMU-RI-TR-01-02, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 2001.

[106] E Simoncelli and J Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. In *Fifth IEEE Int'l Conf on Image Proc*, volume I, Chicago, 4-7 1998. IEEE Computer Society.

[107] A. Singhal, J. Luo, and W. Zhu. Probabilistic spatial context models for scene content understanding. *CVPR*, 2003.

[108] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering object categories in image collections. In *Proceedings of the International Conference on Computer Vision*, 2005.

[109] P. Soille. *Morphological Image Analysis*. Springer, 2003.

[110] T.M. Strat and M.A. Fischler. Context-based vision: Recognizing objects using information from both 2-d and 3-d imagery. *Pattern Analysis and Machine Vision*, 13(10):1050–1065, October 1991.

[111] Erik B. Sudderth, Antonio Torralba, William T. Freeman, and Alan S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 2*, pages 1331–1338, Washington, DC, USA, 2005. IEEE Computer Society.

[112] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381(6582):520–522, 1996.

[113] S. Thrun. Is learning the n-th thing any easier than learning the first? In *NIPS 8*, 1996.

[114] Frank Tong and Ken Nakayama. Robust representations for faces: Evidence from visual search. *Journal of Experimental Psychology: Human Perception and Performance*, 1990.

[115] A. Torralba and P. Sinha. Detecting Faces in Impoverished Images. Face recognition under varying pose. Cbcl paper 208 / ai memo 2001-028, Massachusetts Institute of Technology, Cambridge, MA, 2001.

[116] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. Context-based vision system for place and object recognition. *IEEE Intl. Conference on Computer Vision (ICCV)*, October 2003.

[117] A. Torralba, K.P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, 2004.

[118] A. Torralba and A. Oliva. Statistics of natural image categories. *Network: computation in neural systems*, 14:391–412, 2003.

[119] Antonio Torralba. Contextual priming for object detection. *International Journal of Computer Vision (IJCV)*, 53(2):161–191, 2004.

[120] A. Treisman. Feature binding, attention and object perception. *Philos. Trans. R. Soc. Lond. B. Biol. Sci.*, 353(1373):1295–306, August 1998.

[121] E Ullman, S. & Sali. Object classification using a fragment-based representation. *Biologically Motivated Computer Vision, First IEEE International Workshop*, pages 73–87, May 2000.

[122] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermdediate complexity and their use in classification. *Nature Neuroscience*, 5(7):682–687, 2002.

[123] Shimon Ullman, Erez Sali, and Michel Vidal-Naquet. A fragment-based approach to object representation and classification. In *IWVF-4: Proceedings of the 4th International Workshop on Visual Form*, pages 85–102, London, UK, 2001. Springer-Verlag.

[124] M. Usher, Y. Bonneh, D. Sagi, and M. Herrmann. Mechanisms for spatial integration in visual detection: a model based on lateral interactions. *Spat Vision.*, 1999.

[125] V Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1999.

[126] Paul Viola and Michael Jones. Robust real-time object detection. *IJCV*, 2002.

[127] Johan Wagemans. Detection of visual symmetries. *Spatial Vision*, 9:9, 1995.

[128] M. Weber, W. Welling, and P. Perona. Towards automatic dscovery of object categories. In *CVPR*, 2000.

[129] M. Weber, W. Welling, and P. Perona. Unsupervised learning of models of recognition. In *European Conference on Computer Vision*, volume 2, pages 1001–108, 2000.

[130] L. Wolf and S. M. Bileschi. A critical view of context. *International Journal of Computer Vision (IJCV)*, accepted 2006.

[131] L. Wolf, S. M. Bileschi, and E. Meyers. Perception strategies in hierarchical vision systems. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[132] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[133] T. Zielke, M. Brauckmann, and W. von Seelen. Intensity and edge-based symmetry detection applied to car-following. In G. Sandini, editor, *Computer Vision: European Conference on Computer Vision 1992, (Lecture Notes in Computer Science 588)*, pages 865–873. Springer-Verlag, 1992.