A Statistical Approach to 3D Object Detection Applied to Faces and Cars

Henry Schneiderman

CMU-RI-TR-00-06

May 10, 2000

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Takeo Kanade, Carnegie Mellon, Chair
Alex Pentland, MIT Media Lab
Tai Sing Lee, Carnegie Mellon
Dean Pomerleau, AssistWare Technology

Abstract

In this thesis, we describe a statistical method for 3D object detection. In this method, we decompose the 3D geometry of each object into a small number of viewpoints. For each viewpoint, we construct a decision rule that determines if the object is present at that specific orientation. Each decision rule uses the statistics of both object appearance and "non-object" visual appearance. We represent each set of statistics using a product of histograms. Each histogram represents the joint statistics of a subset of wavelet coefficients and their position on the object. Our approach is to use many such histograms representing a wide variety of visual attributes. Using this method, we have developed the first algorithm that can reliably detect faces that vary from frontal view to full profile view and the first algorithm that can reliably detect cars over a wide range of viewpoints.

Acknowledgments

I would like to thank my advisor, Takeo Kanade, for his guidance in this research. His ideas, insights, suggestions, questions, and enthusiasm were great help in stimulating my thought and in bringing this research forward. I would like to thank the other members of my dissertation committee Tai Sing Lee, Dean Pomerleau, and Sandy Pentland for their helpful feedback and suggestions. I would like to thank John Krumm for originally guiding me into this area of research as part my summer internship in his laboratory in Sandia National Laboratories in 1995. I would like to thank Henry Rowley for his feedback on my research and for providing much useful data and programs. I would also like to thank the many people with whom I had valuable discussions, provided helpful feedback on my research, and provided technical assistance including Chris Lee, Alan Lipton, Rahul Sukthankar, Tim Doebler, Larry Ray, Farhana Kagalwala, David LaRose, Michael Nechyba, Frank Dellaert, Mark Ollis, Simon Baker, Teck Khim Ng, Chris Diehl, Dennis Strewlow, Martin Martin, Dongmei Zhang, YingLi Tian, Bob Collins, Jeff Cohn, Terence Sim, Dave Duggins, Tom Drayer, Yanxi Liu, Tsuhan Chen, Tommy Poggio, Huang Fu Jie, Steve Seitz, Sundar Vedula, Jeff Schneider, Geoff Gordon, and Marina Meila. And finally, most of all, I would like to thank Laura and the rest of my family for their unwavering support and encouragement during my graduate studies.

1 Introduction **1**

1.1 Challenges in Object Detection 2

1.2 Choosing a Representation for Object Detection 4

1.3 Representational Choices for Object Detection. 5

1.4 Detection of Human Faces and Cars 7

1.5 Overview of Approach 7

2 View-Based Detectors **10**

3 Functional Form of Detector: Statistical Representation Using Histograms **13**

3.1 Representation of Object and Non-Object Statistics 14

3.2 Representation of Statistics Using Histograms 15

3.3 Multiple Histograms to Represent Multiple Attributes of Visual Appearance 17

3.4 "Parts-based" Decomposition of Appearance 19

3.4.1 Frequency/Scale Decomposition 19
3.4.2 Orientation Decomposition 21
3.4.3 Generalized Decomposition 21
3.4.4 Spatial Decomposition 21
3.4.5 Geometric Decomposition 22

3.5 Implementation of Visual Attributes 23

3.5.1 Wavelet Transform 24
3.5.2 Quantization of Wavelet Coefficients 27
3.5.3 Specification of Visual Attributes 29

3.6 Overall Decision Rule and Summary 32

3.7 Appendix: Wavelet Transform 34

4 Functional Form of Detector:
Re-derivation from an Ideal Form **36**

4.1 Ideal Functional Form of Detector 36

4.2 Generalizations to Ideal Functional Form 37

4.3 Wavelet Transform of Image 42

4.4 Simplifications to Functional Form 42

4.4.1 Statistical Independence 42
4.4.2 Quantization of Wavelet Coefficients 45
4.4.3 Reduced Resolution in Pattern Position 46

4.5 Conclusion 46

5 Training Detectors **47**

5.1 Images of the Object 47

5.1.1 Collection of Original Images 47
5.1.2 Size Normalization and Spatial Alignment 48
5.1.3 Intensity Normalization 49
5.1.4 Synthetic Variation 50

5.2 Non-Object Images 51

5.3 Training Method (I) - Probabilistic Approximation 52

5.4 Training Method (II) - Minimization of Classification Error using AdaBoost 52

6 Implementation of the Detectors **57**

6.1 Exhaustive Search 57

6.2 Searching for Objects at One Size 57

6.3 Re-using Multi-resolution Information 60

6.4 Heuristic Speed-ups 61

6.4.1 Coarse to Fine Search Strategy 61
6.4.2 Adjacent Heuristic 61
6.4.3 Color Heuristics 61

6.5 Performance Time 62

7 Face Detection Performance **63**

7.1 Results in Face Detection 63

7.2 Positional Decomposition of Face Detection 68

7.3 Analysis of Pair-wise Statistical Dependency 68

8 Car Detection Performance **72**

8.1 Results in Car Detection 72

8.2 Positional Decomposition of Car Detection 74

8.3 Analysis of Pair-wise Statistical Dependency 75

9 Review of Other Statistical Detection Methods **77**

9.1 Comparison of Our Approach to Previous Detection / Recognition Methods 77

9.1.1 Local Appearance Versus Global Appearance 77
9.1.2 Sampling of Local Appearance 77
9.1.3 Representation of Geometric Relationships 78
9.1.4 Representation of Local Appearance using Wavelets 78
9.1.5 Representation of Statistics/Discriminant Functions 79
9.1.6 Estimation of Statistics 79

9.2 Summary of Previous Methods for Face Detection 80

9.2.1 Sung and Poggio [18] 80
9.2.2 Rowley, Baluja, and Kanade [14] 81
9.2.3 Osuna, Freund, and Girosi [19] 82
9.2.4 Moghaddam and Pentland [12] 82
9.2.5 Colmenarez and Huang [20] 84
9.2.6 Burl and Perona [26] 86
9.2.7 Roth, Yang, Ahuja [38] 89

9.3 Summary of Previous Methods for Car Detection 90

9.3.1 Rajagopalan, Burlina, Chellappa [44] 90
9.3.2 Papageorgiou, Poggio [83] 90

10 Conclusion **91**

References **95**

Chapter 1. Introduction

Object detection is a big part of our lives. We are constantly looking for and detecting objects: people, streets, buildings, hallways, tables, chairs, desks, sofas, beds, automobiles. Yet it remains a mystery how we perceive objects so accurately and with so little apparent effort. Comprehensive explanations have defied physiologists and psychologists for more than a century.

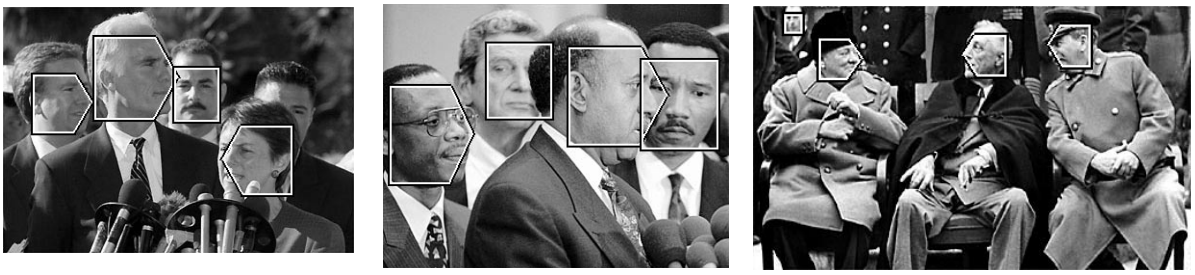


Figure 1. Examples of computer detection of human faces

Figure 2. Examples of computer detection of automobiles

In this thesis, our goal is not to understand how humans perceive, but to create computer methods for automatic object detection. Automated object detection could have many uses. The availability of large digital image collections has grown dramatically in recent years. Corbis estimates it currently has more than 67 million images in its current collection[1]. The Associated Press collects and archives an estimated 1,000 photographs a day[2]. The number of images on the World Wide Web is at least in the hundreds of millions. However, the usability of these collections is limited by a lack of effective retrieval methods. Currently, to find a specific image in such a collection, we have to search using text-based captions and low-level image features such as color and texture. Automatic object detection and recognition could be used to extract more

information from these images and help automatically label and categorize them. By making these databases easier to search, they will become accessible to wider groups of users, such as television broadcasters, law enforcement agencies, medical practitioners, graphic and multimedia designers, book and magazine publishers, journalists, historians, artists, and hobbyists. Automatic object detection could also be useful in photography. As camera technology changes from film to digital capture, cameras will become part optics and part computer (giving true meaning to the term "computer vision"). Such a camera could automatically focus, color balance, and zoom on a specified object of interest, say, a human face. Also, specific object detectors, such as a face detectors and car detectors, have specialized uses. Face detectors are a necessary component in any system for automatic face identification. Car detectors could be used for automatically monitoring traffic.

1.1. Challenges in Object Detection

Automatic object detection is a difficult undertaking. In 30 years of research in computer vision, little progress has been made. The main challenge is the amount of variation in visual appearance. For example, cars vary in size, shape, coloring, and in small details such as the headlights, grill, and tires. An object's orientation and distance from the camera affects its appearance. A more general difficulty is that visual information is ambiguous. Geometric ambiguity exists since the three dimensions of the world are projected on to two in the image. Also, a pixel's intensity depends on many dispersed factors in the environment. It depends on the light sources: their locations, their color, their intensity. It depends on the surrounding objects. Some objects may cast shadows on the object or reflect additional light on to the object. Pixel intensity also depends on the reflective properties of the viewed surfaces. A smooth surface will reflect light differently than a rough one.

Figure 3. Objects of the same class can vary significantly in appearance

Figure 4. Variation due to the pose (the relationship between the orientation of the object and the position of the camera)

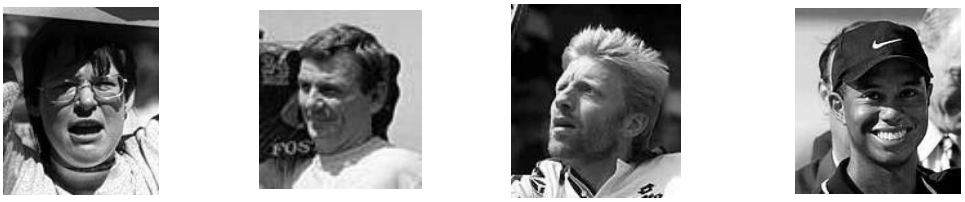


Figure 5. Variation due to lighting and shadowing

Object detection is also difficult because images contain a large amount of data. There may be hundreds or even thousands of input pixels, each of which may carry important information. To use this information to its fullest extent, we would have to build the detector as an enormous table with an entry for every possible input indicating its classification, object or non-object, such as Table 1. Such a representation would account for all the forms of variation we just mentioned. Unfortunately, such a table is infeasible. $256^{400} \approx 10^{963}$ entries would be required for describing the classification of a 20x20 region. Computer power and memory limit us to using a classification rule that is hundreds of orders of magnitude smaller. However, there is hope that we can get by with such a representation. The physical world imposes constraints on the appearance of objects; that is, of all the possible images that could conceivably exist in the physical world, only a small subset actually do. Moreover, people and animals are living examples things that achieve successful perception within their own computational limits.

Table 1: Ideal but infeasible classifier

(1,1)	(1,2)	. . .	(20,20)	classification
0	0	. . .	0	Non-object
0	0	. . .	1	Non-object

Table 1: Ideal but infeasible classifier

(1,1)	(1,2)	. . .	(20,20)	classification
. . .	. . .	. . .	. . .	. . .
35	45	. . .	28	Object
. . .	. . .	. . .	. . .	. . .
255	255	. . .	255	Non-object

1.2. Choosing a Representation for Object Detection

The main issue in object detection is choosing a good representation within our computing constraints. An object detector must accommodate all the variation we mentioned and still distinguish the object from any other visual scene in the world.

The most attractive way to choose a representation would be to use an automated method that would choose it for us. Such a goal has motivated many methods such as genetic algorithms and prune and grow methods for artificial neural networks. However, with current computer power, this is too great a challenge. The search space of possible models is enormous for the high dimensional spaces created by images.

Ultimately, we must make representational choices by hand.[1] In making these choices, we have to ask the question: *What relationships help distinguish the object from the rest of the visual world*? The best we can do is make educated guesses. We can look at the work of other researchers in object recognition and detection and try to understand why some models have been more successful than others. We can look at psychological and biological studies on how animals and people see. And, we can use our intuition about what makes an object distinct from everything else. These sources of knowledge helped us to design the object detector we use in this thesis.

Let us first look at what some of the representational choices and trade-offs are.

---

1. This statement may sound surprising. Most papers on recognition do not discuss making such decisions. This omission is probably because these methods start from a representational framework (e.g., a mixture of Gaussians, multilayer perceptron, quadratic filter, etc.) in which representational possibilities have already been pruned.

1.3. Representational Choices for Object Detection

Below we outline some of the major differences among existing methods for detection and recognition:

• 3D vs. 2D - An object can be represented directly through a 3D model or indirectly through a collection of 2D models. For example, alignment methods [3] use explicit 3D models relating the position of key features. Modular viewpoint-based models [4][30][29] decompose appearance into separate 2D models corresponding to different viewpoints. There can be a small finite number of viewpoints [4] or the 2D information could be represented as a continuous function of the 3D pose of the object [30][29]. Generally, it is easier to represent appearance in 2D models than in 3D models. Conversely, shape can be directly represented using 3D models but only indirectly represented using 2D models.

• Wholistic appearance vs. appearance of parts - There are many algorithms that treat the object's appearance as one monolithic quantity such as [30][29][5][6]. At the other extreme there are algorithms that represent the object's appearance as a collections of parts (edges, corners, complex features) [9][10][39]. The trade-off is that while monolithic models capture relationships across the full extent of the object, describing the model in terms of smaller parts concentrates the representation power and describes the appearance of these smaller portions in greater detail.

• Feature vs. Geometry - If we use a parts-based description, there is trade-off between how much representational power we allocate to the description of the parts themselves and how much we allocate to a description of their geometric relationships. At one extreme, some representations do not model spatial relationships [9][84]. At the other extreme, others use a minimalist feature representation and emphasize geometric relationships [32][41][45][46][47].

• Sampling density - If we represent the object in terms of parts, there is the question of what parts to represent. Do we choose a few salient features. Say the eyes, nose, and mouth on a face or some other relatively small subset of the visual information [26][4][40]? Do we use an "interest operator" or some other automatic method to select a small set of features to use [81][39][82]. Or do we uniformly sample the appearance of the object. Obviously, both representation power

cost increase by using a higher sampling density.

• Non-object representation - Detection can be thought of as a two category classification problem. How do we represent the "non-object" category. Do we not represent it all and thereby implicitly assume a uniform distribution. Or do we model it explicitly using real-imagery.[14][18] And if so what and how do we select images and weight them?

Probabilistic representation - At one extreme we can represent the object as a fixed deterministic template. Such a representation will work well if the object's appearance does not vary and environment (lighting, etc.) is controlled. As object recognition methods try to cope with increasing degrees of variation in the object and environment, they use different modeling techniques. For example, if the only form of variation is lighting intensity, normalized correlation may be the best solution. If the object's appearance is also subject to variation there are a plethora of methods and techniques. Below we discuss some the issues in making these choices:

- Dimensionality reduction - To use any statistical representation any method must significantly reduce the dimensionality of the image in order to avoid overfitting. There are many ways to do this such as linear projection methods[5][29][30], and resolution reduction methods[14][18], or simply by hand selecting features.

- Probability functions vs. discriminant functions - We can model the statistics of each class separately, or we can model the classification problem directly through a discriminant function.

- Flexibility of model - How many variable parameters are fit to the training data? Can the representation divide input space with a hyperplane, conic section[19][81][82], etc. Can the probability distributions represent multiple modes?

- Parametric vs. non parametric - Non parametric models make no assumptions about the nature of the distribution but are more costly in terms of data reduction requirements and retrieval of probabilities.

- Estimation \ learning method- Method for fitting model to training data. Can best fit be achieved or is method subject to local minima. Does closed-form solution exist or is method iter-

ative?

-Complexity of evaluation of probability - Given an input how much computation is required to retrieve the associated probability?

1.4. Detection of Human Faces and Cars

Our representational choices will depend on the object we are trying to detect. It is unlikely that one representation will work for all kinds of objects. In this thesis, we develop methods for detection of human faces and cars. For automobile detection we detect 2-door and 4-door passenger cars excluding all trucks, sport utility vehicles, vans, mini-vans, station wagons, racing cars, and pre-1950 models. For both objects we detect upright objects allowing for variation between full frontal view to full side view (no back views) and we have placed no restrictions on background scenery, lighting, or any other naturally occurring environmental conditions.

We choose faces and cars because we wanted the challenge of two objects that differ in many ways. Faces are natural objects that are irregular in their shape. Cars are man-made and consist primarily of flat surfaces. The surfaces on faces tend to be dull and non-reflective while those on cars tend to be shiny and reflective. Yet, in spite of these difference, we also choose these objects because they are similar in a more abstract way. They are both semi-rigid objects that have distinctive features in a fixed geometric arrangement. On faces, the eyes, nose, and mouth form a unique configuration. On cars, the grill, windshield, tires, and headlights are in a relatively fixed arrangement. This similarity between cars and faces allows us to use the same underlying model for both objects. By being able to do so, we view this work as a first step toward developing a general method that applies to the whole class of semi-rigid objects. Hopefully, with little modification other than that of the training data, we will be able to apply this method to the detection of pedestrians, airplanes, ships, boats, bicycles, tanks, fire hydrants, stop lights, some animals ,and many other objects with similar properties.

1.5. Overview of Approach

We use a 2D view-based approach for representing the 3D geometry of each object. In this approach, we use several decision rules for each object where each rule decides if the object is

present at a particular orientation with respect to the camera. In Chapter 2, we describe how we partition the 3D geometry of each object into decision rules for separate views and how we normalize the size and alignment of the object within each of these views.

All of these decision rules (for different objects and for different viewpoints for the same object) have the same underlying functional form. The only difference among the rules is that each one is trained on a different set of images. The key issue we address in this thesis is the choice of the decision rule's functional form. This function form is the embodiment of all our representational choices.

We use a statistically based representations to model variation in appearance for the object and variation in the appearance of the rest of the visual world. We capture these statistics using a representation based on multiple histograms. Each histogram captures the statistics of a different attribute of visual appearance. These attributes represent appearance over differing spatial extents, frequency ranges, and orientations. We use a large set of different attributes to take advantage of as much information as possible in making a detection decision. Depending on the image, we may have to rely on different cues. For example, in Figure 6 we show the variety of cues that may or may not exist in images of a human face, such as lack or presence of low/high frequencies, glasses, facial hair, shadowing, surface markings, etc. In Chapter 3, we describe these representational choices and derive the functional form of our detector.

Figure 6. The cues available for detecting faces vary widely from image to image.

We also view our representation as our best attempt at approximating an ideal functional form within our computational constraints. In Chapter 4 we re-derive our functional form by a series of

approximations to this ideal form. Through this analysis we gain a better understanding of several issues that are not obvious in the first derivation. This derivation reveals the statistical independence assumptions between the attributes. It also reveals why it is useful to select non-object samples by bootstrapping and why we should explicitly train the detector to reduce the classification error over the training set.

In Chapter 5 we describe how we train each decision rule by collecting statistics from a specified set of training images. First, we describe how we use bootstrapping to select "non-object" samples. We then describe how we use the AdaBoost algorithm to train the decision rules to explicitly minimize classification error on the training set.

In Chapter 6 we describe the implementation of the detector. We discuss issues related to speed and efficiency. In particular, our approach is to search the image in a coarse-to-fine fashion pruning out unpromising candidates along the way.

We describe our performance for face detection in Chapter 7 and our performance for car detection in Chapter 8.

In Chapter 9 we compare our approach to other probabilistic approaches to object detection, specifically those for face detection and car detection.

In Chapter 10 we summarize our approach and discuss future topics of research.

Chapter 2. View-Based Detectors

Our overall goal is to be able to detect the object over a range of orientations, sizes, and positions in an image. We use a 2D view-based approach to accommodate variation in orientation and we use exhaustive search in position and scale to accommodate variation in size and position.

A view-based approach works as follows. For each object, we build several detectors where each one is specialized to specific orientation of the object and can accommodate small amounts of variation around this orientation. To be able to detect an object at any orientation we apply all these detectors to the image and merge their results such that they are spatially consistent. In Figure 7 we show such a face detection result using this approach. In this example, each detector

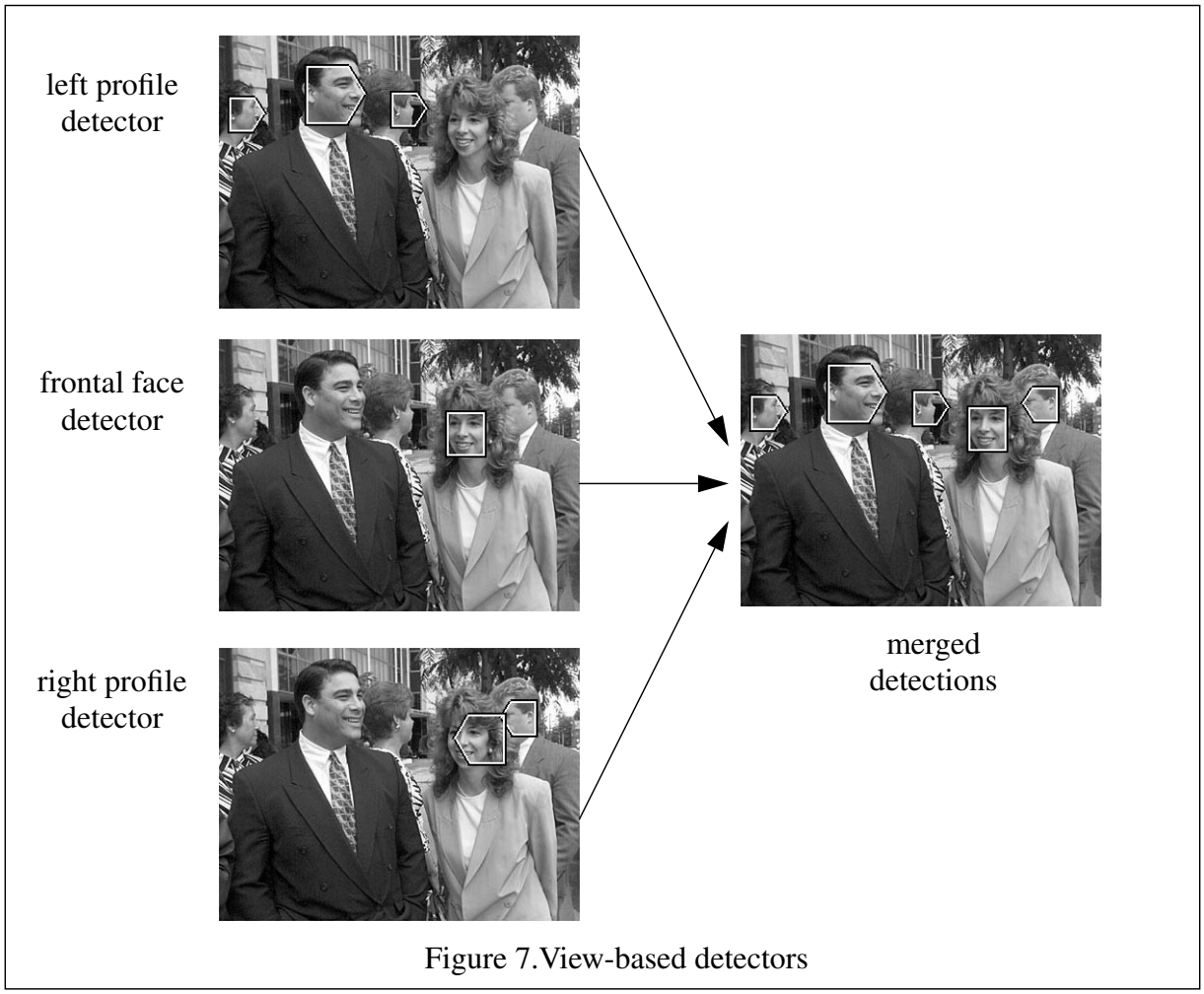


Figure 7.View-based detectors

detects all the faces corresponding to its orientation. The woman in front was initially detected by both the frontal and left profile detectors, because the orientation of her face is somewhat in-between these orientations. However, when the algorithm spatially integrates these results and chooses the more confident detection which in this case is the frontal detection.

It may seem counter-intuitive to use a 2D based model such as this to represent a 3D object, but there is an advantage to doing so. The problem with a 3D model is that we do not have explicit knowledge of the 3D geometry of the object. All our information is in the form of 2D images. To maintain a 3D representation we would have to rely on 3D recovery methods which are error-prone. By maintaining 2D models we avoid introducing such errors into our representation.

The question of how many and which viewpoints to use is an open question. One possible answer is to select viewpoints from aspect graphs if the object has well-defined surfaces [62]. However, our approach was to simply determine the number of viewpoints through experimentation. For face detection, we found that three separate detectors was sufficient: left-profile, frontal views, and right-profile. In practice, we built only two detectors, right-profile and frontal, since we can detect left-profiles by applying the right profile detector to a mirror-reversed image. We show example training images for these in Figure 8. For automobile detection, we originally used

Figure 8. Example training images for frontal and right profile face views

three detectors, left-side, front, and right-side, but found it was necessary to use more. There are several explanations for this. Automobile photographs tend to be taken from a wider variety of vantages, from road level to views from a higher vantage point. In comparison, we usually photograph faces at eye level, except in surveillance cameras. Also, the shape of an automobile is rectilinear. Small changes in angle will produce bigger changes in appearance than they do for a sphere. Overall, we used 15 decision rules corresponding to the following orientations: one fron-

tal viewpoint and 14 side viewpoints. Here again, we only had to train 8 detectors (7 right side detectors and one frontal detector), since 7 viewpoints are mirror reflections of each other. In Figure 9 we show example training images for each of the viewpoint we trained on . We do not detect back views of cars. Also for both object we do not represent in-plane rotations. Both faces and cars tend to appear as upright objects.

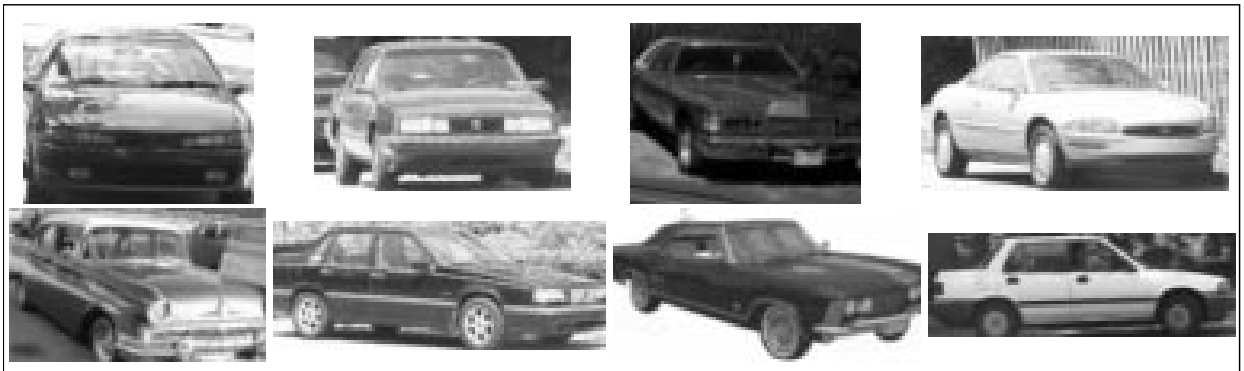


Figure 9. Example training images for each car viewpoint

In addition to detecting the object over variation in orientation, we also have to detect it over variation in size and position within the image. Our approach to detecting the object under these variations is to use exhaustive search. We train each view-based detector to only find the object when it is normalized in size and centered in a given rectangular image window. (We design each such detector to accommodate small variation about this size and alignment.) To then detect the object at position in an image, we have to re-apply each detector at all possible positions of the rectangular window. Then to detect the object at any size, we have to repeat this process over magnified and contracted versions of the original image.

Chapter 3. Functional Form of Detector: Statistical Representation Using Histograms

In this chapter we derive the functional form of our detector. We use a statistical representation to model variation in visual appearance. We model both the statistics of appearance of the object and the statistics of the rest of the world. The difficulty in modeling these distributions is that we do not know their true characteristics. We do not know if they are Gaussian, Poisson, multi-modal, etc. These characteristics are unknown since it is not tractable to analyze the joint statistics of large numbers of pixels. Therefore, we sought statistical models that avoid making strong assumptions about distributional structure while still retaining good properties for estimation and retrieval. As we will explain, the best compromise we found was histograms.

Histograms, however, have one fundamental limitation. A histogram can only use a discrete number of values to describe appearance. More importantly, because of limited computer memory and finite training data, a histogram can only use a relatively small number of discrete values. To overcome this limitation we will describe how we use multiple histograms where each histogram represents the statistical behavior of a different group of quantized wavelet coefficients. With this representation, each histogram represents a different attribute of appearance in terms of spatial extent, frequency range, orientation. Our approach is to use many such histograms to make up for the limited scope and resolution of each individual one.

In this approach, by modeling groups of wavelet coefficients, we capture the statistics of appearance over limited spatial extents. However, we would also like to capture the overall geometric configuration of the object. Therefore, as we will explain, in each histogram, we represent the joint statistics of appearance and position, where we measure position with respect to a local coordinate frame affixed to the object. This representation implicitly captures each part's relative position with respect to all the others.

3.1. Representation of Object and Non-Object Statistics

Coping with variation in appearance is a big problem in object detection. We not only have to cope with variation in the object, but we also have to cope with variation in the rest of the world; that is, we have to distinguish the object from anything else that might occur in the world. For this reason, we model both the statistics of appearance of the object, $P(\textit{image} \mid \text{object})$, and the statistics of appearance of the rest of the visual world, $P(\textit{image} \mid \text{non-object})$.

Intuitively, the non-object statistics help us account for the distinctiveness or uniqueness of features. Some features on an object are more distinctive than others. For example, a paisley pattern is more distinctive than a plain white pattern. On a human face, the eyes are more distinctive than the areas on the cheeks. Cheeks have little texture and could look like many other things in the visual world. However, an "eye-like pattern" and a "cheek-like pattern" will have similar probabilities of occurrence on a face:

$$P(\text{eye-like pattern} \mid \text{face}) \approx P(\text{cheek-like pattern} \mid \text{face}) \tag{1}$$

since each face has two cheeks and two eyes. Yet, an eye seems to be a better indicator of a face and should carry more influence than a cheek in making the classification decision.

We use the statistics of the rest of the world, $P(\textit{image} \mid \text{non-object})$, to capture this notion of distinctiveness. Since "eye-like patterns" are rarer than "cheek-like patterns" in the visual world at large, we would expect that:

$$P(\text{eye-like pattern} \mid \text{non-face}) < P(\text{cheek-like pattern} \mid \text{non-face}) \tag{2}$$

Therefore, we can get a better measure of a given feature's classification power by dividing[1] its object probability by its non-object probability:

---

1. It may seem arbitrary that we divided the two probability distributions in equations (3) and (4). Conceivably, we could have combined them in other ways, such as by subtraction. When we re-derive the decision rule in Chapter 4, we will see how this equation follows from Bayes decision rule.

$$\frac{P(feature|\text{ object})}{P(feature|\text{ non-object})} \tag{3}$$

This ratio will be large for distinctive features on the object -- patterns that occur frequently on the object but occur infrequently in the rest of the visual world.

Our overall model for the decision rule will then be the ratio of the two probabilities:

$$\frac{P(image|\text{ object})}{P(image|\text{ non-object})} > \lambda \tag{4}$$

If this ratio is greater than λ, we will decide that the object is present. If it is less than λ then we decide that the object is not present.

In the rest of the chapter we explain how we represent the two probability distributions, $P(image \mid \text{object})$ and $P(image \mid \text{non-object})$.

3.2. Representation of Statistics Using Histograms

The difficulty in modeling $P(image \mid \text{object})$ and $P(image \mid \text{non-object})$, is that we do not know the true statistical characteristics of appearance either for the object or for the rest of the world. For example, we do not know if the true distributions are Gaussian, Poisson, multimodal. These properties are unknown since it is not tractable to analyze the joint statistics of large numbers of pixels.

Since we do not know the true structure of these distributions, the safest approach is to choose models that are flexible and can accommodate a wide range of structure. The most flexible representations are non-parametric methods, such as nearest neighbor, Parzen windows and other kernel density models. However, there are two problems with non-parametric models. First, the high dimensional nature of images presents a problem. An enormous set of training examples and drastic dimensionality reduction are necessary to avoid overfitting. The second problem is the high computational cost of probability retrieval from these models. For a given input, probability retrieval involves a computation over the entire training set; that is, we have to compare the input to all the training data. Even if efficient strategies such as k-d trees are used, probability retrieval

calculations will be very time consuming.

An alternative would be to use flexible parametric distributions such as a multilayer perceptron (artificial neural network) or mixture model. Each of these models has some flexibility to model multi-modal distributions and become a universal approximators as their sizes increase towards infinity (i.e., a multilayer perceptron with infinite hidden units, or a mixture model with infinite number of modes). However, there are no closed form solutions for fitting these models to a set of training examples. Instead, we must estimate their parameters by iterative procedures: gradient descent (backpropagation) and E-M, respectively. These parameter estimates can become trapped in a local minimum and be suboptimal. Also, it is questionable whether such models are appropriate for detection. For example, a multilayer perceptron forms decision boundaries that are combinations of hyperplanes to a first approximation. It is not clear whether such decision boundaries will be good for separating two or more classes in a high dimensional space [51][52].

Instead of these approaches, we use histograms as a basis for probabilistic representation (see [9][10][54] for other histogram-based methods in computer vision). Histograms are almost as flexible as non-parametric methods. Histograms, however, have the advantage that probabilities can be retrieved by a table look-up rather than a computation over the original training data. Estimation of a histogram is also trivial. We count how often each attribute value occurs in the training data. This involves just one pass through the training images. The resulting estimates are statistically optimal: maximum likelihood, no bias, consistent, and efficient (they satisfy the Cramer-Rao lower bound).

We can model the accuracy of a histogram by considering each bin to be a binary variable: either the input falls in the bin outside of the bin. Let us say the true probability that the input falls in the bin is p. We can then model the probability of counting k occurrences in n trials by a binomial distribution:

$$Pr(\text{k occurences in n trials}) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k} \tag{5}$$

Given values for k and n, we can compute the maximum likelihood estimate for p as:

$$\hat{p}_{ml} = \frac{k}{n} \tag{6}$$

We can also compute the variance in this estimate:

$$E[(p - \hat{p}_{ml})^2] = \frac{p(1-p)}{n} \tag{7}$$

As we can see the variance in our statistical estimates will be quite low if n is large. This formula will tell us how accurate our probability estimates are for given amount of training data, n.

3.3. Multiple Histograms to Represent Multiple Attributes of Visual Appearance

The main drawback of a histogram is we can only use a relatively small number of discrete values to describe appearance. For example, if we were to directly represent the appearance as the set of pixel values over a region, say just a small 8x8 region, there would be $256^8 \sim 10^{19}$ bins in the histogram. It is impossible to represent a histogram this large in computer memory. In general histogram size will be limited by a combination of memory and limited amounts of training data.

For the sake of argument, let us say that histogram size is limited to 100,000. With this size we are effectively limited to a 16.6 bit representation of appearance since $2^{16.6} \sim 100{,}000$. Clearly, 16.6 bits of information is not adequate for the purposes of object detection. We would like a richer description of appearance. To do so, we will extract *multiple attributes* from the image and compute their histograms separately. Each attribute will represent something different about the image, for example, low frequency information over large spatial extents, high frequency information over small extents, etc. We will define these later in this chapter.

First, let us see how this strategy can improve our representation of local appearance beyond 16.6 bits. For the sake of illustration, let us consider the representation of an 8x8 region, X. If each pixel is 8 bits, the total region requires 512 bits for exact representation.

Let us assume a maximum of 100,000 total histogram bins. As mentioned above, if we use one visual attribute, we are effectively representing only 16.6 bits of the image:

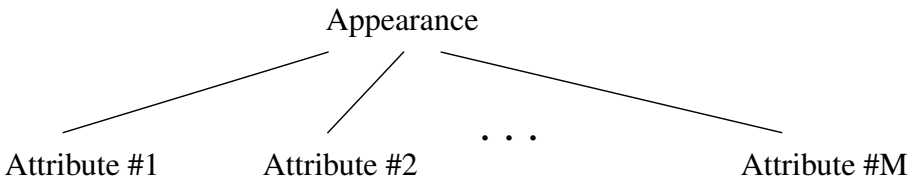


Figure 10.Decomposition of appearance into multiple attributes

$$pattern_{\mathrm{r}} = \mathrm{Quantize}_{\mathrm{r}}(X) \qquad pattern_1 \in (r_1, r_2, \ldots, r_{100,000}) \quad \sim 16.6 \text{ bits} \tag{8}$$

where we use the variable $pattern_r$ to denote the range of discrete values: $r_1, \ldots r_{100,000}$.

The representation of probability then becomes:

$$P(X) \approx P_{\mathrm{r}}(pattern_{\mathrm{r}}) \tag{9}$$

However, if we use two attributes each represented by 50,000 discrete values, then we are effectively representing the region with a total of 31.2 bits:

$$\begin{aligned} pattern_{\mathrm{p}} &= \mathrm{Quantize}_{\mathrm{p}}(X) \qquad pattern_{\mathrm{p}} \in (p_1, p_2, \ldots, p_{50,000}) \quad \sim 15.6 \text{ bits} \\ pattern_{\mathrm{q}} &= \mathrm{Quantize}_{\mathrm{q}}(X) \qquad pattern_{\mathrm{q}} \in (q_1, q_2, \ldots, q_{50,000}) \quad \sim 15.6 \text{ bits} \end{aligned} \tag{10}$$

Our representation of probability then becomes:

$$P(X) \approx P_{\mathrm{p}}(pattern_{\mathrm{p}}) P_{\mathrm{q}}(pattern_{\mathrm{q}}) \tag{11}$$

where we have assumed statistical independence between the $pattern_1$ and $pattern_2$. We will explore this assumption in more detail in Chapter 4.

Of course, we could use more attributes, each with fewer discrete values:

$$
\begin{aligned}
pattern_{\mathrm{s,1}} &= \mathrm{Quantize}_{\mathrm{s,1}}(X) && pattern_{\mathrm{s,1}} \in (s_{1,1}, s_{1,2}, \ldots, s_{1,m}) \\
pattern_{\mathrm{s,2}} &= \mathrm{Quantize}_{\mathrm{s,2}}(X) && pattern_{\mathrm{s,2}} \in (s_{2,1}, s_{2,2}, \ldots, s_{2,m}) \\
& && \ldots \\
pattern_{\mathrm{s,n}} &= \mathrm{Quantize}_{\mathrm{s,n}}(X) && pattern_{\mathrm{s,n}} \in (s_{n,1}, \ldots, s_{n,m})
\end{aligned}
\tag{12}
$$

Our representation of probability then becomes:

$$
P(X) \approx \prod_{k=1}^{n} P_{\mathrm{s,k}}(pattern_{\mathrm{s,k}}) \tag{13}
$$

In the extreme, we could even represent the region using 50,000 binary-valued attributes. Such a representation could be thought of as a large decision tree.

As we can see, the number of bits used to represent appearance can be increased as we decompose our representation using more attributes. However, the visual representation cannot be effectively increased beyond 512 bits, since that's how many bits are in the original representation of the region.

While this strategy improves the fidelity of the visual representation, the larger goal is to improve the accuracy of the probability, $P(X)$ given by equation (9). We would expect that equations (11) and (13) can improve the accuracy over equation (9). However, many questions remain. How do we decompose appearance into different attributes? How many attributes do we need? And how many discrete values do we use for each attribute? There are no conclusive answers to these questions. In the next sections we provide some provisional answers. We begin by discussing how we decompose appearance into different attributes.

3.4. "Parts-based" Decomposition of Appearance

In this section, we describe the different ways we decompose appearance to form different visual attributes. In particular, we decompose the image in frequency, orientation, space, and geometry.

3.4.1. Frequency/Scale Decomposition

In general, we would like to base our representation on visual attributes that are suited to the

scale of the features on the object. If we base a representation solely on attributes that are too small -- say, one pixel at the extreme -- they will not be powerful enough to describe anything distinctive about the object. (Although there is a method for frontal face detection that works surprisingly well using such a pixel-based representation [38]). On the other hand, we do not want a representation that only describes the object over large spatial extents. First, large attributes are not a desirable allocation of modeling resources. They spread the limited representational power over the entire object. This is true of any representation that reduces dimensionality in some way. For example, if we look at the most significant eigenimages of a human face, we will notice a lack of detail for small characteristics, such as the eyes and the nose. Second, a large template is known to be sensitive to small differences in scale, position, and orientation [55]. Finally, the matching of large regions may be strongly influenced by "irrelevant" pixels. On many objects, such as a car, there will be large indistinctive areas such as the hood and windshield that are punctuated by relatively smaller areas of distinctive detailing such as the headlights and grill. In matching a large region, the majority of the pixels will come from untextured parts and dominate selection of the match (using any norm that weighs each pixel equally such as L1 or L2). This may be one of the reasons, some researchers have tried to improve the eigenimage approach [5][30] by using templates of smaller size to describe specific features [35][34].

Since important cues will exist at many sizes, the best solution is to describe the object over a range of scales. In the very least, we will need attributes to describe large areas, mid-sized areas, and small areas. Such a representation may seem redundant. Small areas exist as part of large ones. However, we can define these attributes to be completely independent, if we also divide up the frequency content between them. Since low frequencies only exist over large areas and high frequencies can exist over small areas, the most natural decomposition is to use some attributes to represent low frequencies over large spatial extents, attributes to represent mid-range frequencies using mid-sized attributes, and attributes to represent high frequencies over small spatial extents as illustrated by Figure 11.

The attributes describing small areas can do so at high resolution. On a human face, they will be important for representing features such as the eyes, nose, and mouth. Similarly on a car, small attributes are needed for the grill, headlights, tires, and various lines and boundaries.

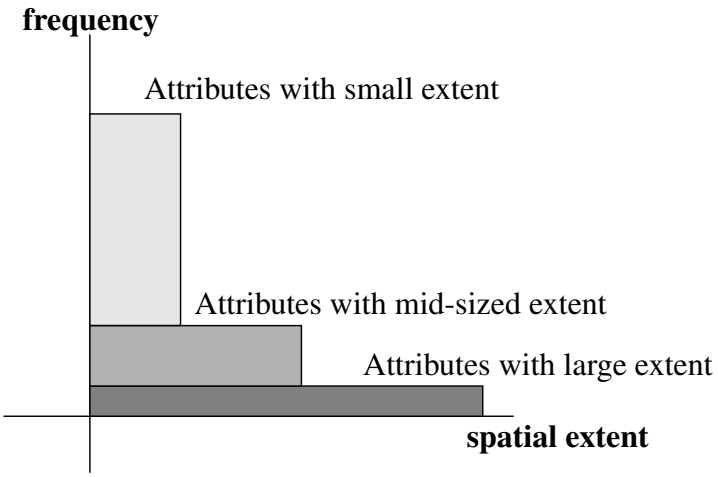


Figure 11. Decomposition of appearance in terms of spatial extent and frequency range

Attributes computed over larger areas will be able to capture many other types of features. On a face, the eye sockets are usually darker than the forehead. On a car, various surfaces such as the hood, windshield, and fenders may differ in average intensity. By having these attributes specialized for low frequency content, we reduce their sensitivity to misalignment.

3.4.2. Orientation Decomposition

In addition to having attributes that are specialized in frequency content, we also specialize some attributes in orientation content. For example, an attribute that is specialized to horizontal features can devote greater representation power to horizontal features than if it also had to describe vertical features.

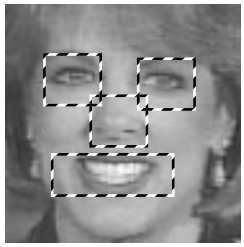
3.4.3. Generalized Decomposition

Finally, important cues may not be strictly localized at one orientation or in one frequency band. Some cues may exist over a wide range of frequencies. For example, edges involve all frequencies. Similarly, not all cues exist at one orientation. Corners involve both horizontal and vertical edges. Therefore, we define additional attributes that span wider ranges of frequencies and/or orientations. The trade-off is that these attributes will be limited to smaller spatial extents.

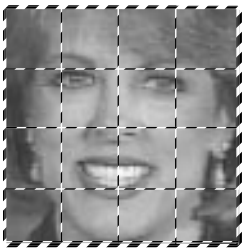
3.4.4. Spatial Decomposition

Since each of the attributes will have a limited spatial extent we will have to spatially decom-

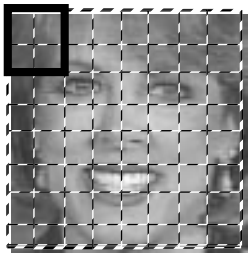
pose the object We can decompose it in many ways. We could sample a few regions that seem distinctive to us or that we determine experimentally as shown in Figure 12a. Alternatively, we could sample at regular intervals over the full extent of the object. For instance, we could choose sampling intervals to achieve critical sampling as shown in Figure 12b.



a. Selective sampling



b. Critical sampling



c. Sampling with overlap

Figure 12. Examples of sampling strategies

Instead, our approach is to oversample each attribute allowing these samples to partly overlap as shown in Figure 12c. We do this for several reasons. First, while salient features, such as the eyes, nose, and mouth on a face, are probably the most important for detection, other important features may exist, too. By sampling everywhere, we include anything that might contribute to the detection decision. We also improve our detection decision by using more measurements. This idea is similar to using more data to improve the accuracy of a statistical estimate [56].

Sampling with overlap may appear to lead to greater computational cost for evaluating the attributes. However, it does not. If we were to sample the object more sparsely, we would still end up densely sampling attribute values in each the input image. The reason for this is that we have to search the input image at all possible object locations, including overlapping ones. For each of these locations we collect a different set of attribute values. Since we densely sample these object locations we will end up with a dense sampling of attribute values, as well.

3.4.5. Geometric Decomposition

By decomposing and sampling the object spatially, we lose the ability to represent all the relationships between the various parts. We believe that the spatial relationships of these various parts is a very important cue for detection. On the human face, the eyes, nose, and mouth appear

in a fixed geometric configuration. This arrangement of these features is a distinctive characteristic of faces. If this geometric information becomes scrambled, the face becomes unintelligible as shown in Figure 13.

Figure 13. Face with geometric relationships scrambled

To model geometry, we represent the positions of each sample with respect to a coordinate frame affixed to the object as shown in Figure 14. This representation captures each sample's relative position with respect to all the others and implicitly captures many geometric properties [57]. For each attribute, we then represent the joint probability of appearance and position for both the object and the rest of the world: $P(pattern(x,y), x, y \mid \text{object})$ and $P(pattern(x,y), x, y \mid \text{non-object})$ where $pattern(x, y)$ is the discrete value of the specified attribute sampled at position x, y.

However, there is no need to represent position at the original resolution of the image. Instead, we represent position at a coarser resolution to save on modeling cost and to implicitly accommodate small variations in geometric arrangements of the parts of an object. The functions $i(x)$ and $j(y)$ indicate this reduction in spatial resolution, where r will be either 4, 8, 16, depending on the specific attribute:

$$i(x) = \left[\frac{x}{r}\right] \qquad j(y) = \left[\frac{y}{r}\right] \tag{14}$$

3.5. Implementation of Visual Attributes

In this section we describe the actual attributes we compute based on the image decomposition

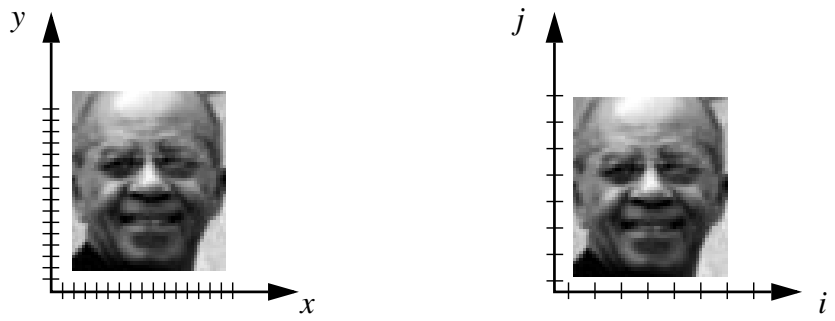


Figure 14. Coordinate frames affixed to object.
Coarse resolution in attribute position in terms of *i* and *j*

described in the last section. We choose each attribute to represent a subset of quantized wavelet coefficients. Overall, we use 17 different attributes with variation in spatial extent, frequency range, orientation, and quantization.

3.5.1. Wavelet Transform

To form different attributes, we would like to decompose the image in the ways described in the last section. In particular, we would like a representation jointly localized in space, frequency and orientation. To do so, we perform a wavelet transform on the image. The wavelet transform organizes the image into subbands that are localized in orientation and frequency. Within each subband, each coefficient is spatially localized as well.

The wavelet transform is not the only method that can produce a representation that is jointly localized in space, frequency, and orientation. Both the short-term Fourier transform (also known as "Gabor wavelets") and pyramid algorithms can create representations that are jointly localized in space, frequency and orientation. Wavelets, however, have no redundancy. The number of transform coefficients is equal to the original number of pixels in the image. These other multi-resolution decompositions produce transforms that are larger than the original image.

We use a three level transform with a biorthogonal 5/3 wavelet filterbank[1] [58], producing 10 subbands, as shown in Figures 15 and 16. Level 1 coefficients describe the lowest octave of fre-

1. Low-pass coefficients: -0.1768, 0.3535, 1.0607, 0.3535, -0.1768. High-pass coefficients: 0.3535, -0.7071, 0.3535.

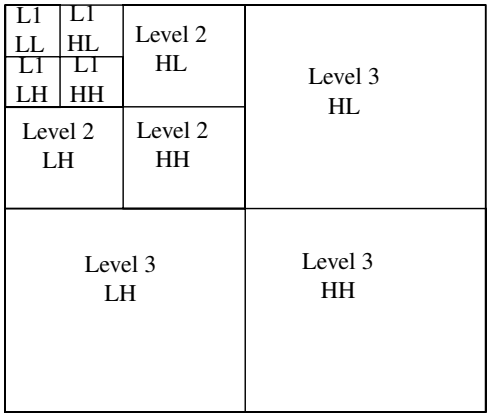


Figure 15. Wavelet representation of an image

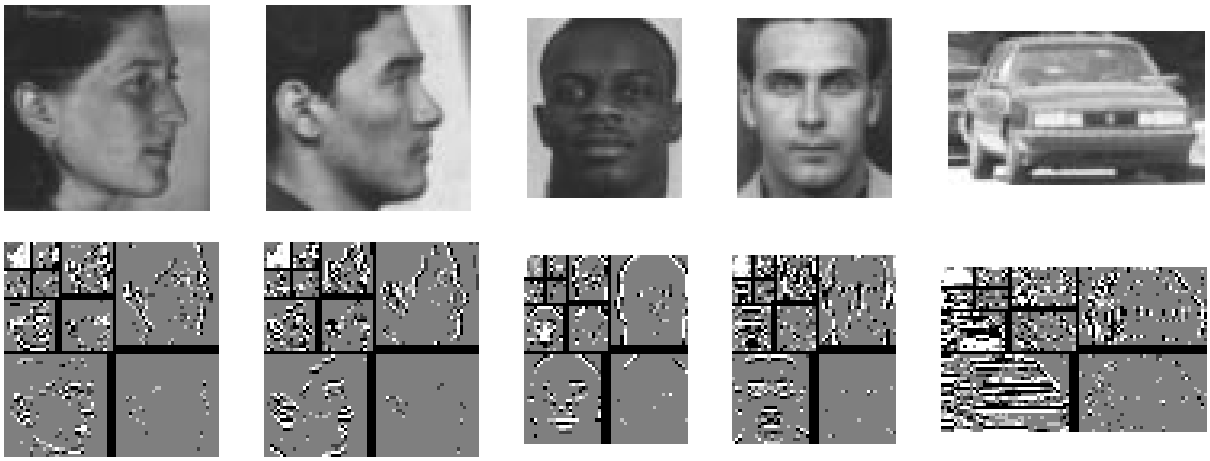


Figure 16. Images and their wavelet transforms.
Note: the wavelet coefficients are each quantized to five values.

quencies. Each subsequent level represents a higher octave of frequencies. In terms of spatial extent, a coefficient in level 1 describes four times the area of a coefficient in level 2, which describes four times the area of a coefficient in level 3. In Figure 17 we illustrate these relationships.

In terms of orientation, the LH bands are the result of low-pass filtering in the horizontal direction and high pass filtering in the vertical direction giving horizontal features. Similarly, HL represents vertical features.

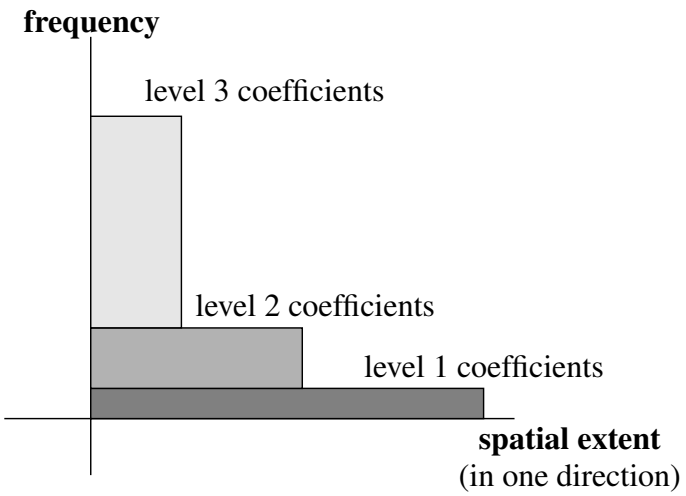


Figure 17. Properties of wavelet coefficients

An important property of the 5/3 filter is that its high-pass filter has one vanishing moment. An nth order vanishing moment means:

$$\sum_{j=0}^{m-1} j^n d(j) = 0 \tag{15}$$

for high pass filter coefficients *d*(*0*) through *d*(*m-1*). A zeroth vanishing moment indicates that the coefficients add to zero. The filter is "blind" to constants and the DC level of the signal across the filter's extent. The output of the filter only represents changes in intensity. Only level 1 LL represents DC information since it is the result of low-pass filtering in both directions. Many high-pass filters used for compression are designed to have a first and 2nd vanishing moments also. This can be accomplished by using longer filters. These high-pass filters are then "blind" to linear and quadratic changes in the image. It remains an open question, whether there are any advantage to doing this for detection.

For a more complete description of the transform and the properties of a biorthogonal 5/3 filter bank see the Appendix of this chapter, Section 3.7, and [58][59].

3.5.2. Quantization of Wavelet Coefficients

Since we plan to represent the statistics of each attribute using a histogram, we must quantize the wavelet coefficients so they each take on a finite range of values. In particular, we chose to limit each attribute to 10,000 discrete values. We made this choice out of practical considerations. To avoid excessive storage and memory we would like to keep overall histogram size under 10^6 bins. Position in x,y will take on order 10^2 values, therefore, we limit attributes to 10^4 values.

There are conflicting goals in choosing a quantization resolution for the transform coefficients. We need enough quantization levels to preserve the recognizability of the object. If the image reconstructed from the quantized transform is unrecognizable to a human, it is unlikely to have enough information for computer recognition. Yet we want to use as few quantization levels as possible. By doing so, we can represent larger spatial extents with a fewer number of discrete values.

First, let us explain how we represent the various subbands. To begin with, we discard all HH bands. This reduces the amount of information by almost a third while degrading appearance little. In Figure 18 we show images with and without HH subbands.

Figure 18. Original images on top, HH bands removed on bottom.

We represent the level 1 LL subband differently from the remaining subbands. It is the result of successive low-pass filterings in both directions. It represents a very low resolution copy of the original image. All the other subbands are produced by a high pass filtering in one direction. Since the coefficients in the high pass filter sum to zero, its output represents differential measurements. To make the level 1 LL band consistent, we represent each coefficient in terms of its dif-

ferences with its neighbors. In particular, we generate two representations for each coefficient. In the "vertical" representation each coefficient is replaced by the difference between it and its right most neighbor. In the "horizontal" representation, each coefficient is replaced by the difference between it and its top neighbor. This captures horizontal features. These two representations could be thought of analogous to a HL and LH band respectively, and we effectively treat them as such.

We would like to quantize all coefficients to 5 levels. Fewer than 5 levels were not sufficient for preserving appearance, especially for cars. In Figure 19 we show images reconstructed from these quantized subbands for both 3 and 5 quantization levels:

Figure 19. Quantization of HL and LH bands: Original images on top, 5 levels per coefficient in middle, 3 levels per coefficient on bottom. Note: level 1 LL band is not quantized.

The problem with 5 quantization levels is that it restricts us to attributes based on a very small number of coefficients. 6 coefficients per attribute would exceed our limit of 10,000 discrete values since 5^6 = 15,625. To overcome this limitation, our strategy is to apply only 3 quantization levels each time we sample a coefficient. However, since each coefficient will be sampled multiple times, (because of sampling with overlap and samples from multiple attributes) we can apply a different set of quantization thresholds each time we sample. This way, each coefficient can be evaluated with respect to 5 (or more) levels but as part of different samples. More specifically, to

achieve 5 quantization levels we will need 4 quantization boundaries, $\{-q_2, -q_1, q_1, q_2\}$. We break these quantization boundaries into two sets: $\{-q_1, q_1\}$ and $(-q_2, q_2\}$. We will then specify each attribute by a window of sampling sites with specified quantization thresholds. For example, an attribute defined over a 2x2 neighborhood could be defined with the following quantization at each site:

$-q_1$, q_1	$-q_2$, q_2
$-q_2$, q_2	$-q_1$, q_1

Figure 20. Quantization levels assigned to a 2x2 neighborhood

By doing so, a coefficient quantized with respect to $\{-q_1, q_1\}$ for one sample will be quantized with respect to $(-q_2, q_2\}$ for another sample shifted by one. With this strategy, we choose attributes computed over windows of 8 coefficients, where each attribute is described by 3^8 = 6,561 discrete values.

3.5.3. Specification of Visual Attributes

We define four broad types of visual attributes: *intra-subband*, *inter-frequency*, and *inter-orientation*. The first type of attribute takes all 8 coefficients from one subband. We call these *intra-subband* attributes. These attributes are the most localized in frequency and orientation. *Inter-frequency* attributes use coefficients from multiple frequency bands but the same orientation band. *Inter-orientation* attributes use coefficients from both orientations, LH and HL, but from within the same frequency band. Finally, an *inter-orientation* / *inter-frequency* attribute has no restrictions on which the bands from which it draws is coefficients [61]. In Figure 21 we illustrate these differences in the support of each type of attribute.

Our assumption is that all of these relationships are important. Some visual cues will only involve coefficients with a subband, others involve coefficients over a range of frequencies, some involve both horizontal and vertical components, and some as relations across both frequency and orientation. Our provisional modeling choice was to give these relationships approximately equal representational power. With some slight exceptions, we choose to represent each transform coef-

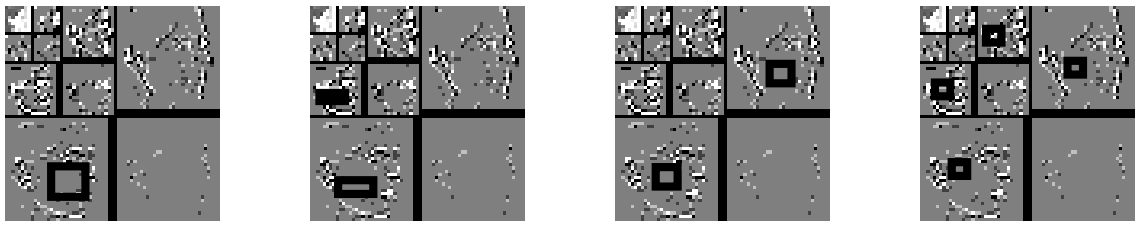


Figure 21. Regions of support for each type of attribute

ficient in one intra-subband attribute, one inter-orientation attribute, at least one inter-frequency attribute, and one inter-orientation / inter-frequency attribute.

Overall, we used total of 17 attributes for each face detector and 13 attributes for each car detector. We also experimented with 33 attributes for two of the car detectors, where the 20 additional attributes represented long, straight and narrow spatial extents. These additional attributes improved performance, but the memory requirements were too large to do further development. Below we describe the 17 attributes we used for face detection including the 13 attributes we used for car detection. (We sample each of these attributes at small step sizes across the extent of the image, i.e, wavelet transform)

• 6 intra-subband attributes. These are taken from the following subbands as listed below in Table 2. Each is computed from a set of 8 coefficients, a 3x3 block with one coefficient removed. In Figure 22 we show the shape of this window and for each coefficient we indicate the quantization boundaries it uses (similar to diagram in Figure 20).

Table 2: Intra-subband Attributes

Subband	
Level 1 LH	faces, cars
Level 1 HL	faces, cars
Level 2 LH	faces, cars
Level 2 HL	faces, cars
Level 3 LH	faces, cars

Table 2: Intra-subband Attributes

Subband	
Level 3 HL	faces, cars

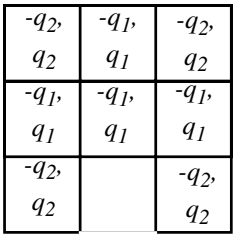
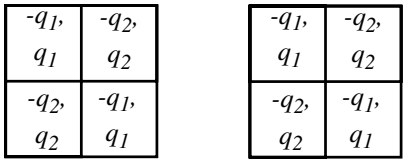


Figure 22. Region of support and quantization for intra-subband attributes

• 4 inter-orientation attributes. These are taken from the following subbands as listed below in Table 3. Each of these is computed from spatially corresponding 2x2 blocks in the two subbands as shown in Figure 23

Table 3: Inter-orientation subbands

Subbands	
Level 1 LL (horizontal) Level 1 LL (vertical)	faces
Level 1 LH, Level 1 HL	faces, cars
Level 2 LH, Level 2 HL	faces, cars
Level 3 LH, Level 3 HL	faces, cars



LH or LL horizontal

HL or LL vertical

Figure 23. Quantization for inter-orientation subbands

• 6 inter-frequency attributes as given in Table 4

Table 4: Inter-frequency subbands

subbands	
Level 1 LL (horizontal), Level 1 LH	faces
Level 1 LH, Level 2 LH	faces, cars
Level 2 LH, Level 3 LH	faces, cars
Level 1 LL (vertical), Level 1 HL	faces
Level 1 HL, Level 2 HL	faces, cars
Level 2 HL, Level 3 HL	faces, cars

• 1 inter-orientation / inter-frequency attribute for faces only involving 1 LL-horizontal, level 1 LL-vertical, level 1 LH , level 1HL, level 2 LH, level 2 HL.

With this representation, the attributes that use level 1 coefficients will cover the largest spatial extents and the lowest and smallest range of frequencies. Conversely, the attributes that use level 3 coefficients will cover the smallest spatial extents and the highest (and largest) range of frequencies. The attributes that use level 2 coefficients represent attributes that are intermediate in spatial extent and frequency range. As we will see in Chapter 6, this representation allows us to search for objects using a coarse to fine heuristic strategy. For example, we can evaluate the image using only attributes that consist of level 1 coefficients. Based on the output probabilities from these attributes we can decide how to evaluate the rest of the image. We can terminate the search at those regions that have low probability and continue searching for the object in those regions that have higher probability.

3.6. Overall Decision Rule and Summary

By combining all the representational decisions in this chapter we can write the overall functional form of our decision rule as:

$$\frac{P(image|\text{ object })}{P(image|\text{ non-object })} \approx \prod_{k}^{17} \prod_{x,\, y \in \text{ region}} \frac{P_k(pattern_k(x, y), i(x), j(y)|\text{ object })}{P_k(pattern_k(x, y), i(x), j(y)|\text{ non-object })} > \lambda \quad (16)$$

Let us explain how we combine the various ideas in this chapter to get this form:

1. Object and Non-Object probability - As described in Section 3.1, we write the probability function of the object in the numerator and the probability function of the "non-object" in the denominator.

2. Histograms - We represent $P_k(pattern_k(x,y),\ i(x),\ j(y)\ |$ object) and $P_k(pattern_k(x,y),\ i(x),\ j(y)\ |$ non-object), as histograms. Each of these histograms represents the joint statistics of appearance, given by $pattern_k$, and position on the object, given by $x,\ y$. We compute multiple histograms over 17 discrete-valued visual attributes, $pattern_k$.

3. Visual attributes - Each visual attribute, $pattern_k$, represents the values of a subset of quantized wavelet coefficients. We use four types of visual attributes:

   a. Intra-subband - Coefficient subset is localized in frequency, orientation and space.

   b. Inter-frequency - Coefficient subset is localized in orientation and space only.

   c. Inter-orientation - Coefficient subset localized in frequency and space only.

   d. Inter-orientation / inter-frequency - Coefficient subset localized only in space.

4. Sampling with overlap - We sample each attribute at regularly spaced overlapping intervals within each candidate object region.

5. Multiplication of probabilities - In Chapter 4 we will see why we multiply the probabilities corresponding to different observations. In particular, we will see how doing so corresponds to an assumption of statistical independence of the observations.

6. Threshold - We have not explained how to set the threshold, λ. In practice we will select it

empirically. We can, however, interpret it as the ratio of the prior probabilities, if the probability models are accurate (which they are not) as we will also see in Chapter 4.

3.7. Appendix: Wavelet Transform

The wavelet transform decomposes an image into "subbands" that are localized in frequency and orientation. A wavelet transform is created by passing the image through a series of filter bank stages. In Figure 24 we show one stage in such a series. In each stage we filter the image first in the horizontal direction. We use a one-dimensional high-pass and low-pass filter pair. The high-pass filter (wavelet function) and the low-pass filter (scaling function) are finite impulse response filters. In other words, the output at each point depends only on a finite portion of the input. The filtered outputs are then downsampled by a factor of 2 in the horizontal direction. These signals are then each filtered by an identical filter pair in the vertical direction. We end up with a decomposition of the image into 4 subbands denoted by LL, HL, LH, HH. Each of these subbands can be thought of as a smaller version of the image representing different image properties. The LL (low-pass in both horizontal and vertical) is simply a low-frequency, low-resolution version of the original. The LH (high-pass filtering in the vertical direction and low-pass filtering in the horizontal) subband represents vertical features, such as lines and edges. Similarly, HL represents horizontal features. And, to some extent HH represents diagonal features.

These four subbands could be thought of as one frequency band or one level in a wavelet transform. To further decompose the image in frequency, we iterate on the LL band; that is we decompose the LL band just the same way we decomposed the original image by sending it through another stage of filtering identical to the first. We can do this multiple times with each new LL band to generate multiple levels in the decomposition. In our representation we use a 3 stage filter bank generating a a 3-level decomposition in frequency.

If the high pass - low pass filter pair is chosen properly, the original image can be reconstructed from its transform with no loss of information. Such filter banks are called perfect reconstruction filter banks. Several books describe their design [58][59]. There are several filter pairs we can choose that will give perfect reconstruction. We can choose orthogonal filters of various lengths: Daubechies 4, Daubechies 6, Daubechies 8 or we could choose linear phase of various lengths 3

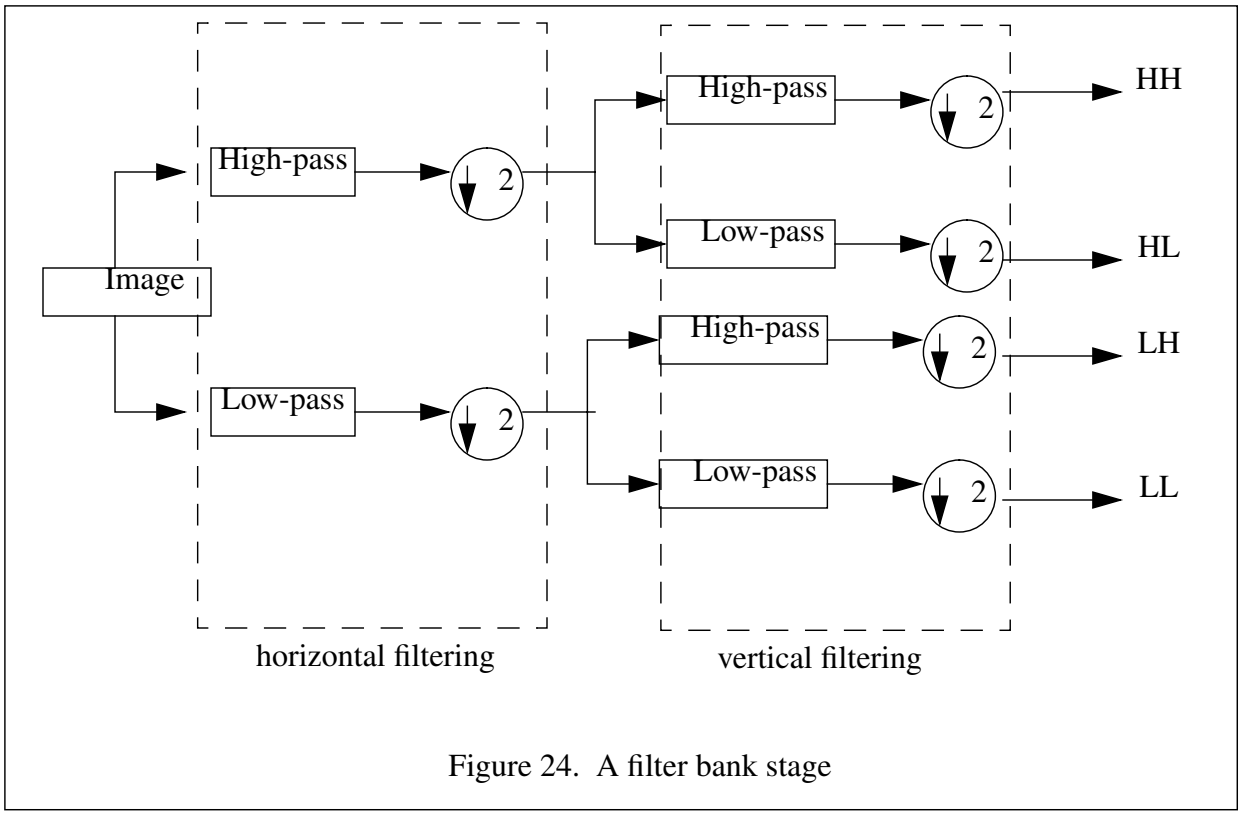


Figure 24. A filter bank stage

(low-pass)/5(high-pass), 7/9, FBI 7/9, 11/13, etc. It is not possible to choose filters that are both orthogonal and linear phase except for the trivial case of Haar wavelets. We choose symmetric 5/3 mainly because it served several practical needs. By having linear phase there is no ambiguity about the alignment of the coefficients in successive levels of the decomposition. We choose filters of short length, 5/3 rather than 7/9 or 11/13 because we wanted to capture localized information. Remember, each coefficient describes a localized region in space, frequency and orientation. Also, of course, smaller filters have less computational cost. Finally, there is also some evidence that human vision is less sensitive to errors with symmetric filters (i.e. linear phase) [60], but there has not been rigorous verification of this.

Chapter 4. Functional Form of Detector: Re-derivation from an Ideal Form

We can also view the functional form of our detector as representing our best attempt at approximating to an ideal functional form within our computational constraints. In this chapter, we re-derive our functional form through a series of approximations to an ideal functional form. By deriving our decision rule this way we get a clear picture of the functional form's representational capacity and its deficiencies. In particular, we have a complete record of all the transformations and simplifications that limit its representational power. Also, through this analysis, we gain a better understanding of several issues that were not apparent in the first derivation in Chapter 3. This derivation reveals why it is useful to select non-object samples by bootstrapping and why we should train the detector to explicitly reduce the classification error on the training set. Also, we see why we divide object probability by non-object probability (in section 3.1) -- a consequence of Bayes' decision rule. Similarly, we see how multiplying the probabilities from different attributes (Section 3.6) corresponds to an assumption of statistical independence of the observations.

4.1. Ideal Functional Form of Detector

In Chapter 1 we introduced an ideal classifier as a large table as shown in Table 5 below. This classifier is ideal in several ways. First, it based on a full representation of the data. The classifier uses the entire raw input, not a selected or filtered portion of it. Second, it minimizes the probability of error, assuming each entry in the table is labeled with its most probable classification. Finally, it is concise representation of the output. For each input, we simply represent its classification and nothing else. Of course, such a table is not feasible. It is not possible to enumerate every possible input in a table. There are too many; we would need a table with 256^{400} entries for classification of a 20x20 region where each pixel takes on 256 values.

Although this decision rule is intractable, it provides a useful starting point for deriving a feasible decision rule. In our case, the functional form of our decision rule, equation (17), can be

Table 5: Ideal but infeasible classifier

256^{20x20}

(1,1)	(1,2)	. . .	(20,20)	classification
0	0	. . .	0	Non-object
0	0	. . .	1	Non-object
. . .	. . .	. . .	. . .	. . .
35	45	. . .	28	Object
. . .	. . .	. . .	. . .	. . .
255	255	. . .	255	Non-object

derived by a series of two generalizations (Section 4.2), a wavelet transform (Section 4.3), and three simplifications (Section 4.4), as we will explain below.

$$\prod_{k}^{17} \prod_{x,\, y \in \text{ region}} \frac{P_k(pattern_k(x, y), i(x), j(y) \mid \text{ object })}{P_k(pattern_k(x, y), i(x), j(y) \mid \text{ non-object })} > \lambda \tag{17}$$

4.2. Generalizations to Ideal Functional Form

We immediately notice several differences between equation (17) and the ideal model, Table 5. First, in the ideal classifier, each entry in the table states a classification, object or not-object. On the other hand, our classifier outputs a continuous value which it compares to a threshold. Also, our classifier separately represents object and non-object properties, whereas the table does not. Thus, to transform the table into equation (17), we must first make it more general in these ways. These generalizations turn out to have important implications for the training of our detector, as we will show.

First, we generalize the output of this table from binary values (object, non-object) to posterior probabilities, as shown in Table 6. The posterior probability, $P(\text{object} \mid \textit{image})$, states the probability that object is present for a given input. The probability that the object is not in the image is simply the complement of the posterior probability, $P(\text{non-object} \mid \textit{image}) = 1.0 - P(\text{object} \mid \textit{image})$. To make the best decision, we always choose the classification, object or non-object, which has the higher probability. Equivalently, we decide that the object is present in the input

Table 6: Ideal classifier using posterior probabilities

(1,1)	(1,2)	. . .	(20,20)	P(Object | *Image*)
0	0	. . .	0	0.000001
0	0	. . .	1	0.000003
. . .	. . .	. . .	. . .	. . .
35	45	. . .	28	0.87521
. . .	. . .	. . .	. . .	. . .
255	255	. . .	255	0.00004

image if:

$$P(\text{object}|image) > 0.5 \tag{18}$$

This decision rule is known by the names of Bayes decision rule and the maximum *a posteriori* (MAP) decision rule.

Now to generalize the decision rule further, we substitute Bayes theorem for P(object | *image*) in equation (18):

$$\frac{P(image|\text{object})P(\text{object})}{P(image|\text{object})P(\text{object}) + P(image|\text{non-object})P(\text{non-object})} > 0.5 \tag{19}$$

Through some algebra we can re-write the decision rule as a likelihood ratio test:

$$\frac{P(image|\text{object})}{P(image|\text{non-object})} > \frac{P(\text{non-object})}{P(\text{object})} = \lambda \tag{20}$$

The left side of this equation is called the likelihood ratio. It divides object probability by non-object probability. If the ratio of these two probabilities is greater than the ratio of the prior probabilities, the right side of the equation, then we decide that the object is present. In equation (3) in Chapter 3 we divide object probability by non-object probability as a direct consequence of this

equation.

There are two advantages to writing the Bayes' decision rule as a likelihood ratio test as given by equation (20). First, it is often easier to separately collect statistics for the two probability functions, *P*(*image* | object) and *P*(*image* | non-object), than it is to directly collect statistics for the posterior probability function. Also, by expressing the decision rule this way, we factor out the contribution of the prior probabilities and combine them into a scalar threshold, λ. By changing this threshold we can apply the algorithm to different settings in which the prior probabilities vary.

To use such a representation, we must build a table that stores two values for each input, *P(image* | object*)* and *P(image* | non-object*)*, as shown below in Table 7. There are several conse-

Table 7: Ideal classifier using separate models for object and non-object probabilities

(1,1)	(1,2)	. . .	(20,20)	*P*(*Image* | Object), *P*(*Image* | Non-object)
0	0	. . .	0	0.00000013, 0.013
0	0	. . .	1	0.00000032, 0.014
. . .	. . .	. . .	. . .	. . .
35	45	. . .	28	0.0092, 0.00045
. . .	. . .	. . .	. . .	. . .
255	255	. . .	255	0.00007, 0.03

quences of using a decision rule derived from this generalized form instead of one directly derived from the concise form in Table 5. The generalized form will work just as well if our probability estimates are accurate representations of the true probabilities. However, problems arise if there is some inaccuracy in the modeled probabilities. These errors in the probability will be propagated and lead to errors in our classification decision. The question is: will our probability models for *P*(*image* | object) and *P*(*image* | non-object) be accurate? Unfortunately, the answer is no. First, there will be error in our estimates themselves because they are computed from finite training data. In particular, we would need an extremely broad sample of imagery to get an accurate estimate of *P*(*image* | non-object). An even greater source of error, though, is that our functional

form is derived through incorrect assumptions about the nature of the true distribution. As we will see, in Section 4.4, there are several simplifications we will make to equation (20) in order to achieve a computationally feasible decision. It is unavoidable that some of these simplifications will be inaccurate or partially inaccurate.

We can improve our detection performance if we bypass the intermediate estimates of the probability functions and instead train our detector to directly minimize the classification error. There will be errors in our estimate of the classification function, too, but it is better to estimate this directly rather than combining two sources of error from intermediate estimates. We will later describe how we train the detector to do so using the AdaBoost algorithm in Chapter 5.

The fact that our end goal is classification, not probabilistic modeling, can also guide us in answering an important question: how do we select training data for the *non-object* class? Should we use any imagery that is not of the object or should we be more selective? Let us consider a hypothetical situation. In Figure 25, we plot some hypothetical probability functions, $P(x \mid \text{object})$ and $P(x \mid \text{non-object})$, for a one-dimensional classification problem in terms of a feature, x. We also plot the classification function derived from these probabilities. For input values of x in which one class is overwhelmingly favored over the other, a qualitatively accurate representation of their probabilities is sufficient to be accurate in classification. It is not important to represent all the little bumps and ripples indicated by A, B, and C in Figure 25. However, for values of x where there is not a clear difference between $P(x \mid \text{object})$ and $P(x \mid \text{non-object})$, such as those indicated by D, it is more important to represent the ratio of the probabilities accurately. If we represent the probabilities accurately in this region, the classification boundary will be as precise as possible[15][27]. Therefore, in selecting samples for the non-object class, it is more important to select samples that are more likely to be confused with the object; that is, those close to the decision boundary. (This concept of selecting samples near the decision boundary is similar to the way support vector machines (SVMs) [15][24] work.) In Chapter 5 we will explain how we do this using bootstrapping. Of course, it is still necessary to have a broad sampling of the non-object class, so we will be qualitatively correct in other parts of the input space.

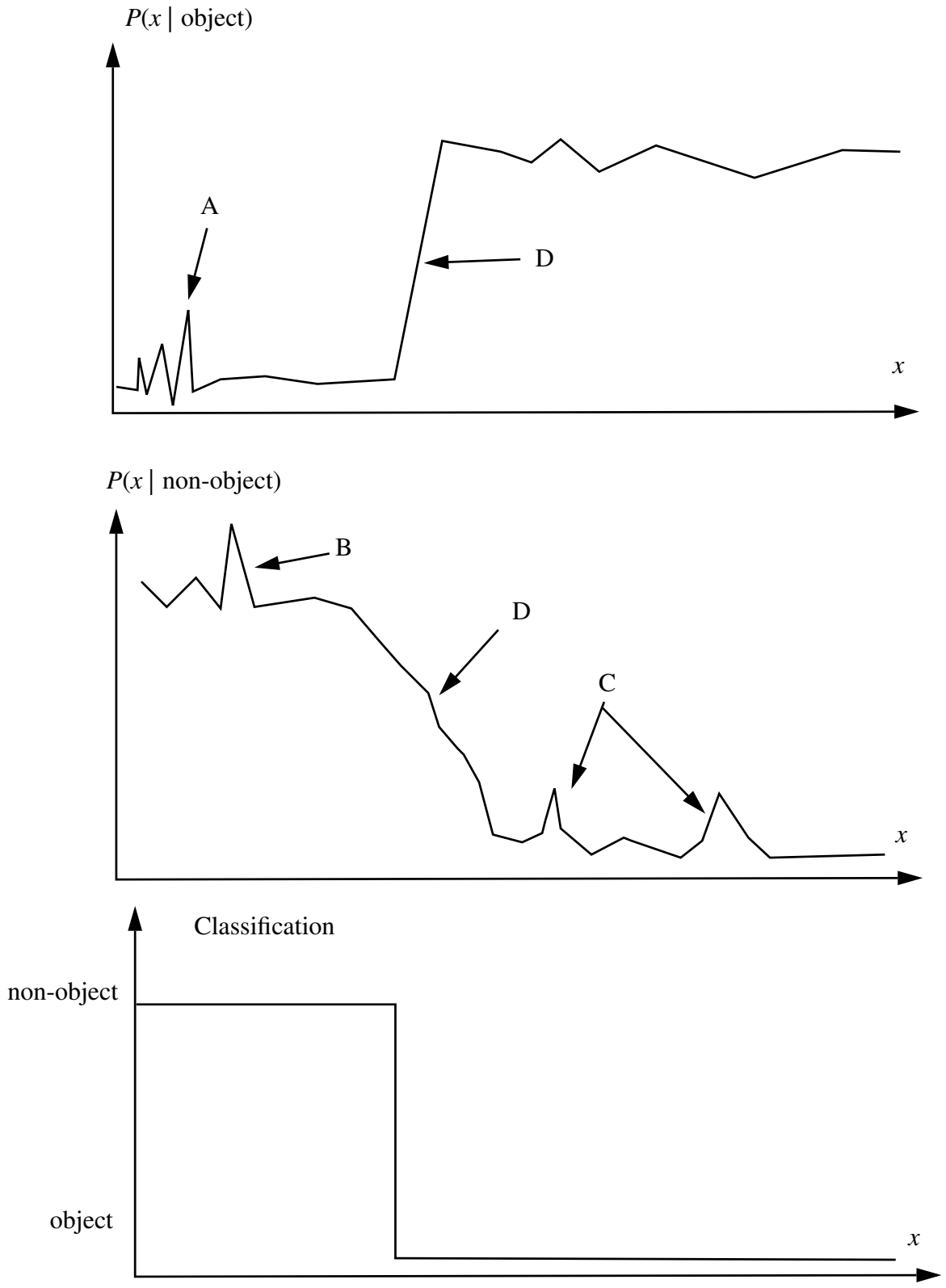


Figure 25. Probability functions and classification function for a hypothetical classification problem (adapted from [27])

4.3. Wavelet Transform of Image

As we described in Section 3.5.1, we perform a wavelet transform of the image. It terms of representational capacity, this transformation has no consequences. The wavelet transformation is fully invertible, so no information is lost. We can represent this transform mathematically as:

$$\frac{P(image|\text{object})}{P(image|\text{non-object})} = \frac{P(W = \text{WT}(image)|\text{object})}{P(W = \text{WT}(image)|\text{non-object})} \tag{21}$$

where W represents the wavelet transform of the image.

4.4. Simplifications to Functional Form

We make three major simplifications to equation (21) in order to get a computationally feasible form for the decision rule.

4.4.1. Statistical Independence

As we have mentioned, computer power and memory limit us to only representing the joint probability of small subsets of wavelet coefficients. We do not model the statistical dependency outside these subsets. By doing so we transform equation (21) in the following way:

$$\frac{P(W|\text{object})}{P(W|\text{non-object})} \approx \frac{P(w_1, \ldots, w_8, x, y|\text{object})\text{P}(w_9, \ldots, w_{16}, x, y|\text{object})\ldots}{P(w_1, \ldots, w_8, x, y|\text{non-object})\text{P}(w_9, \ldots, w_{16}, x, y|\text{non-object})\ldots} \tag{22}$$
$$\frac{\ldots\text{P}(w_{n-7}, \ldots, w_n, x, y|\text{object})}{\ldots\text{P}(w_{n-7}, \ldots, w_n, x, y|\text{non-object})}$$

where $W = (w_1, w_2, \ldots, w_n)$ represents the wavelet transform of the image, and x and y indicate the spatial positions of the group of coefficients with respect to a coordinate frame affixed to the object. (Note: In the representation above, each coefficient is shown as part of only one subset. In practice, we will use each coefficient as part of multiple subsets; that is we will model each coefficients joint statistics with several different groups of coefficients. Each of these subsets will correspond to a different visual attribute modeling statistical dependency over different frequency bands, orientations, or spatial extents (see Section 3.5).) By making this simplification, we

implicitly assume statistical independence between the sets of coefficients. This assumption is the reason we multiplied the probabilities in the overall functional form of our decision rule, equation (16) in Section 3.6.

This statistical independence simplification limits our representational power. In general, we cannot represent relationships among coefficients that are not grouped together into a subset. This includes aspects of appearance that are similar across the full extent of the object, such as skin color or texture on a human face. We cannot represent how the positions of the features systematically vary when viewing perspective changes. We cannot represent long-range symmetries across the object, such as the horizontal symmetry of the human face and cars, where features on one side are strongly correlated with features on the other side, such as the eyes and ears on a human face. We cannot represent the interaction between the lighting and the 3D geometry, i.e., depending on the position of the light source, some parts of the object will be brighter than others. This relationship changes in a systematic fashion as light source position changes.

We can analyze this simplification from several other perspectives. If we take the log of the left side of equation (22), it becomes a sum of log probabilities:

$$\log \frac{P(w_1, \ldots, w_8, x, y \,|\, \text{object}) \ldots \mathrm{P}(w_{n-7}, \ldots, w_n, x, y \,|\, \text{object})}{P(w_1, \ldots, w_8, x, y \,|\, \text{non-object}) \ldots \mathrm{P}(w_{n-7}, \ldots, w_n, x, y \,|\, \text{non-object})} =$$
$$\log \frac{P(w_1, \ldots, w_8, x, y \,|\, \text{object})}{P(w_1, \ldots, w_8, x, y \,|\, \text{non-object})} + \ldots + \log \frac{\mathrm{P}(w_{n-7}, \ldots, w_n, x, y \,|\, \text{object})}{\mathrm{P}(w_{n-7}, \ldots, w_n, x, y \,|\, \text{non-object})} \tag{23}$$

In this form, we can interpret the classifier as a linear discriminator where the log probabilities are weights and the features are binary-valued vectors indicating which patterns have been detected [64]. [17] documents the limitations of linear discriminators and, in particular, linear discriminators cannot represent exclusive-or and parity type functions. However, it is improbable that parity-type functions play much of a role as cues for distinguishing objects. They are the most unstable functions that are mathematically possible; the smallest possible pertubation in the input will lead to a maximum change in the output.

We can also view this equation as a "simple" or "naive" Bayes classifier [63]. It can be shown

that such a model cannot represent *k* of *n* type problems [64]. Similar to parity functions, this type of relationship probably does not play an important role in distinguishing objects.

Nevertheless, it has also been demonstrated that the naive Bayes classifier performs well in a number of classification problems, even when there is significant statistical dependency among the attributes. In particular, it has been shown that it performs equivalent to or better than other concept learners [64]. There is not a full theoretical understanding of why this is true. Although [64] shows that the classifier is optimal for certain problems in which statistical independence does not hold such as conjunctions and disjunctions, they also show it can be optimal for other types of problems, too.

Below we illustrate how classification can be accurate even when the assumption of statistical independence is violated and the statistical representation is inaccurate. Let us consider two random variables, *x* and *y*. Let's say the *x* and *y* are statistically dependent for both object and non-object:

$$\begin{aligned} P(x = f(y)|y, \text{object}) &= 1 \\ P(x = f(y)|y, \text{non-object}) &= 1 \end{aligned} \tag{24}$$

where *f(y)* is a deterministic function.

The correct decision rule is:

$$\begin{aligned} \frac{P(x, y|\text{object})}{P(x, y|\text{non-object})} &= \frac{P(x|y, \text{object})P(y|\text{object})}{P(x|y, \text{non-object})P(y|\text{non-object})} \\ &= \frac{P(y|\text{object})}{P(y|\text{non-object})} > \lambda \end{aligned} \tag{25}$$

The decision rule under the (incorrect) assumption of statistical independence is:

$$\frac{P(x, y|\text{object})}{P(x, y|\text{non-object})} \approx \frac{P(x|\text{object})P(y|\text{object})}{P(x|\text{non-object})P(y|\text{non-object})} =$$

$$= \left(\frac{P(y|\text{object})}{P(y|\text{non-object})}\right)^2 > \gamma \tag{26}$$

However, if we choose $\gamma = \lambda^2$ then the two decision rules become exactly the same even though we have completely violated the assumption of statistical independence in the 2nd decision rule.

In the above example, the assumption of statistical independence worked partially because the joint behavior of x and y was the same for both the object and the non-object class. In general, this assumption will lead to errors if $P(x \mid y, \text{object})$ and $P(x \mid y, \text{non-object})$ are very different functions. This underscores the problem with naive Bayes classifiers or any classifier in general; that is, we will have errors when we do not model important relationships, such as that between x and y when $P(x \mid y, \text{object})$ and $P(x \mid y, \text{non-object})$ are very different functions. No classifier is immune to this problem except the ideal model, Table 5, which is infeasible. In a naive Bayes classifier, the relationships that are not modeled are very explicit. In other models, they may be more hidden (although if we can write such a classifier as a sum or product, we can easily put it in a form equivalent to a naive Bayes classifier). Thus, with any kind of classifier, it is most important to represent the relationships that distinguish the two classes. If we miss important relationships, we may have classification errors.

4.4.2. Quantization of Wavelet Coefficients

In quantizing the wavelet coefficients each subset of wavelet coefficients is replaced by a discrete variable, *pattern*, that takes on 6,561 possible values. Below we factor equation (22) in terms of the 17 attributes we defined in Section 3.5.3 and write the attributes in terms of this discrete valued variable:

$$\prod_{k}^{17} \prod_{x,\, y \in \text{ region}} \frac{P_k(pattern_k(x, y), x, y \mid \text{ object })}{P_k(pattern_k(x, y), x, y \mid \text{ non-object })} > \lambda \tag{27}$$

4.4.3. Reduced Resolution in Pattern Position

We also reduce the resolution of x, y. For level 3 patterns, resolution is reduced by a total factor of 16 (4 in each direction). For level 2, it also reduced by 16. For level 1 it is reduced by a factor of 64 (8 in each direction). We represent this reduction by the functions, $i_k(x)$ and $j_k(y)$.

$$\prod_{k}^{17} \prod_{x,\, y \in \text{ region}} \frac{P_k(pattern_k(x, y), i_k(x), j_k(y) \mid \text{ object })}{P_k(pattern_k(x, y), i_k(x), j_k(y) \mid \text{ non-object })} > \lambda \tag{28}$$

4.5. Conclusion

The derivation of our functional form from an ideal form gives us another perspective on our modeling choices. We see that the main modeling deficiencies are from not modeling the relationships of all wavelet coefficients, by quantizing the values of the wavelet coefficients, and by representing position of the wavelet coefficients at coarse resolution. Therefore, assuming we have adequate training data and an accurate estimate of the classification function, we can attribute any deficiency in performance to these simplifications.

Chapter 5. Training Detectors

So far we have only chosen the *form* of the decision rule; that is, we have specified the number of histograms, the size of each histogram and the variables over which we compute each histogram. We have not specified the actual values within each histogram, $P_k(pattern_k(x,y),\ i(x),\ j(y) \mid \text{object})$ and $P_k(pattern_k(x,y),\ i(x),\ j(y) \mid \text{non-object})$ that are used in the decision rule. We compute these statistical values from various sets of images. This process of gathering statistics is usually referred to as *training*. Specifically, we use a set of images of the object to generate samples for training $P_k(pattern_k(x,y),\ i(x),\ j(y) \mid \text{object})$ and we use images that do not contain the object to train $P_k(pattern_k(x,y),\ i(x),\ j(y) \mid \text{non-object})$. In this chapter, we begin with a discussion of our training images for faces and cars in Section 5.1. In Section 5.2 we discuss the training images we use for the non-object class. Then in section 5.3, we describe a basic training algorithm in which we estimate each histogram separately then in Section 5.4 we describe an alternative training procedure which minimizes the classification error on the training set using the AdaBoost algorithm.

5.1. Images of the Object

In this section we describe the training examples we use for training $P_k(pattern_k(x,y),\ i(x),\ j(y) \mid \text{object})$. We describe where we collected our images, how we align and normalize each training image, how we correct for lighting, and how we generate synthetic variations from each original image.

5.1.1. Collection of Original Images

We gathered face images from a number of sources including the following image collections: FERET[1], NIST mug shot database[2], Harvard's face database[3], and CMU's face collection[4]. We also used many images we found at a variety of sites on the World Wide Web. Overall, we gath-

1. Provided by Jonathon Phillips
2. See http://www.nist.gov/srd/nistsd18.htm.
3. Provided by Woodward Yang
4. Provided by Henry Rowley and Shumeet Baluja

ered about 2,000 images of frontal faces and 2,000 images of profile faces.

We collected car images using our own camera and from the World Wide Web, mostly from car sales sites and car enthusiast sites. The latter sites were a good source of photographs of older cars. Overall, we gathered between 250 and 500 images per viewpoint with a total of over 2,000 images.

5.1.2. Size Normalization and Spatial Alignment

To reduce the amount of variation among each set of training images, we aligned all the training images with respect to a prototype image. We aligned the images using a set of pre-defined feature points that we hand-labeled for each image. Using these feature points, we applied the translation, scaling, and rotation [7] that brought each image into alignment with the prototype.

Below we give the sizes and examples of the images we used for training each detector. Notice that each of these images contains some of the background. We do so because the silhouette of the object against the background is a strong cue for detection for many of these objects, particularly for faces in profile. In Section 5.1.4 we will discuss how we alter the background to generate multiple samples from the same original image.

Figure 26. Face frontal viewpoint. Size = 56x48

Figure 27. Face side viewpoint. Size = 64x64

Figure 32. Car viewpoint #5. Size = 48x80

Figure 28. Car viewpoint #1. Size = 56x72

Figure 29. Car viewpoint #2. Size = 48x80

Figure 30. Car viewpoint #3. Size = 56x72

Figure 31. Car viewpoint #4. Size = 40x88

Figure 33. Car viewpoint #6. Size = 40x104

Figure 34. Car viewpoint #7. Size = 40x96

Figure 35. Car viewpoint #8. Size = 32x96

5.1.3. Intensity Normalization

For faces, to also reduce the variation within each set of training images, we normalized the

intensity of each image. We normalized the left and right sides of each training image separately. For each side we scale all the intensity values by a specified scalars, γ_{left} and γ_{right}:

$$\begin{aligned} I'(x, y) &= \gamma_{left} I(x, y) && \text{if } y < y_o \\ I'(x, y) &= \gamma_{right} I(x, y) && \text{if } y \geq y_o \end{aligned} \tag{29}$$

However, when we sample an attribute that extends across both sides of the object, we normalize the entire sample using the value of γ of the center pixel. We choose these correction factors by hand for each training image. However, for cars we did not normalize the training images.

We normalize the two sides of the face in the training images to compensate for situations in which the object is illuminated unevenly. For example, a face could be brightly illuminated on one side and in shadow on the other as shown below in Figure 36.

Figure 36. Examples of uneven illumination

When we detect the object, for both faces and cars, we evaluate each side of each candidate over a set of 5 discrete values for γ. We then choose the best response from each side and sum them to give the total response for the candidate.

5.1.4. Synthetic Variation

We will improve the accuracy of our object detectors by using more training data. In particular the accuracy of our histogram estimates will increase as we use more training examples (see equation (7) in Section 3.2).

To expand our training set, we generated synthetic variations of each original training image. Depending on the object, we generated between 1600-6400 synthetic variations through small variations in position, orientation, size, aspect ratio, background scenery, lighting intensity, and

frequency content. Overall, this gave us at least 1,000,000 training examples for each object. In order to substitute different background scenery, we segmented the objects from the background in the training images. We segmented these images by hand and by automatic methods when possible. Also, in the synthetic variations, we modified frequency content by using low pass filters with various responses. Below we show several of the forms of synthetic variation we used on each original image:

Figure 37. Synthetic rotational variation

Figure 38. Synthetic variation in aspect ratio and zoom

Figure 39. Synthetic variation in frequency content (low-pass filtering)

5.2. Non-Object Images

We collected these images from photography collections[1] and from the World Wide Web, particularly the Washington University archive[2]. We tried to get a balance of indoor and outdoor scenes, urban, industrial, and rural scenes. Overall, we used over 2,000 such images.

To select non-object samples that resemble the object, we used bootstrapping [18][14]. Bootstrapping is a two-stage process. We first trained a preliminary detector using image samples drawn randomly from the non-object image collection. We then ran this preliminary detection over the entire collection of non-object images. We then selected samples from each of these images where the detector gave a high response and we combined these samples with the original random samples. For some detectors we repeated this process several times gathering more and

1. Provided by John Krumm and Henry Rowley
2. http://www.wuarchive.wustl.edu

more non-object samples. We then used the final combined set of samples for training the final detector.

5.3. Training Method (I) - Probabilistic Approximation

A set of 34 histograms makes up our decision rule: $P_k(pattern_k(x,y),\ i(x),\ j(y) \mid \text{object})$ and $P_k(pattern_k(x,y),\ i(x),\ j(y) \mid \text{non-object})$ for $k = 1 \ldots 17$. To train our car detectors we estimated each histogram separately. To estimate each histogram we simply counted how often each pattern occurs at each position in the appropriate set of training examples. To estimate $P_k(pattern_k(x,y),\ i(x),\ j(y) \mid \text{object})$ we use images of the object, and to estimate $P_k(pattern_k(x,y),\ i(x),\ j(y) \mid \text{non-object})$ we use images that do not contain the object. Occasionally, a pattern and position combination may not occur together in a training set. The corresponding bin in the histogram will have zero count denoting zero probability. However, it may not be desirable to actually assign zero probability to this bin during testing, since its probably of occurrence usually will not be zero. To correct for this situation, we use Laplace correction[8], where we add one to each bin in the histogram.

5.4. Training Method (II) - Minimization of Classification Error using AdaBoost

Training each histogram separately as described in Section 5.3, however, is not optimal for classification. To achieve better results, we should explicitly train our classifier to minimize classification error, as described in Chapter 4. In this section, we describe how we estimate these histograms to minimize classification error using the AdaBoost algorithm. In particular, we trained both the frontal and profile detectors using this approach.

AdaBoost [70][71][72][73]is a method for training a classifier to have low classification error on the training set. Similarly, other methods [74] may work for this purpose as well. AdaBoost can be applied to almost any classification algorithm. It works by training multiple instances of the classifier. Let's call each of these classifiers $h_t(x)$. AdaBoost then takes a weighted sum of these classifiers as the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{30}$$

AdaBoost trains each of the classifiers, $h_t(x)$, by assigning different weighting to the training examples. For the first classifier, $h_t(x)$, all training examples are given equal weight. For all subsequent classifiers, $h_t(x)$, $t > 1$, AdaBoost assigns more weight to training examples that have been incorrectly classified by the previous classifier, $h_{t-1}(x)$. Conversely, it decreases the weight for training examples that were correctly classified.

In using AdaBoost to train our detector we continue to collect each histogram by counting, but, we increment each bin based on the weight assigned to the current training example. We scale and round the training image weights to integers for this purpose.

Following the development of [72], the algorithm works as follows. Assume that we have a sequence of m training examples $\{(x_i, y_i)\}$, where x_i is the input and $y_i = \{-1, +1\}$ is the desired class label. Initialize the weights for each training example as $D_1(i) = 1/m$.

For $t = 1. .T$:

1. Train $h_t(x)$ using $D_t(i)$

2. Compute α_t (we discuss how to compute it below)

3. Update:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t} \tag{31}$$

where Z_t is a normalization factor so that D_{t+1} will be a distribution:

$$\sum_{i=1}^{m} D_{t+1}(i) = 1$$

The final classifier is then given by equation (30) above.

This algorithm achieves the following bound on the training error:

$$\Pr(H(x_i) \neq y_i) \leq \prod_{t=1}^{T} Z_t \tag{32}$$

To understand the power of this bound, let us consider a simple classifier with output {-1, +1} instead of a continuous value, as in our classifier. Then

$$Z_t = \sum_i D_t(i)\left(\frac{1+u_i}{2}e^{-\alpha_t} + \frac{1-u_i}{2}e^{\alpha_t}\right) \tag{33}$$

where $u_i = y_i h_t(x_i)$.

We can then choose α as to minimize the right side of this equation:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right) \qquad r_t = \sum_i D_t(i)u_i \tag{34}$$

Plugging this choice back into Z gives:

$$Z_t = \sqrt{1-r_t^2} \tag{35}$$

Since this is guaranteed to be less than one, we can see how the classification error can be driven towards zero after a few iterations.

In our case, the classifier, $h_t(x)$, outputs a continuous value. Therefore, we cannot determine α_t analytically to minimize Z_t as we have done in equation (34). Instead, we minimize Z_t numerically in terms of α_t, since Z_t is strictly quadratic (see [72] for proof). In this form it is more difficult to put a bound on the error rate of the classifier. However, we would expect the same behavior to hold. For instance, [72] gives an upper bound on the error when the output of the classifier is continuous but bounded.

In using AdaBoost to train our detector we continue to collect each histogram by counting, but, we increment each bin based on the weight assigned to the current training example. We scale and round the training image weights to integers for this purpose. However, we train each histogram multiple times for multiple iterations of the algorithm. For each training example we increment the appropriate bins by the weight assigned to the training example.

It should also be noted that forming the final classifier in equation (30) does not actually increase the complexity of the classifier. Each classifier is of the form

$$h_t(z_i) = \sum_{k=1}^{17} \sum_{x,\, y \in \text{ region}} \log \frac{P_{k,\,t}(pattern_k(x, y), i(x), j(y) \mid \text{ object })}{P_{k,\,t}(pattern_k(x, y), i(x), j(y) \mid \text{ non-object })} \tag{36}$$

In this form, we only need to store one value for each discrete combination of p_o, i_o, and j_o to represent the quantity:

$$\log \frac{P_{k,\,t}(pattern_k(x, y) = p_o, i(x) = i_o, j(y) = j_o \mid \text{ object })}{P_{k,\,t}(pattern_k(x, y) = p_o, i(x) = i_o, j(y) = j_o \mid \text{ non-object })} \tag{37}$$

Since the composite classifier is:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t \sum_{k=1}^{17} \sum_{x,\, y \in \text{ region}} \log \frac{P_{k,\,t}(pattern_k(x, y), i(x), j(y) \mid \text{ object })}{P_{k,\,t}(pattern_k(x, y), i(x), j(y) \mid \text{ non-object })} \right) \tag{38}$$

we also can store one value for each discrete combination of p_o, i_o, and j_o given by:

$$\sum_{t=1}^{T} \alpha_t \log \frac{P_{k,t}(pattern_k(x, y) = p_o, i(x) = i_o, j(y) = j_o \mid \text{object})}{P_{k,t}(pattern_k(x, y) = p_o, i(x) = i_o, j(y) = j_o \mid \text{non-object})} \tag{39}$$

An unresolved issue in using AdaBoost is when to stop the iteration. Since weighting favors the most difficult samples, those close to the decision boundary, we may become susceptible to overfitting. Our approach was to monitor the performance of the algorithm, empirically using a cross-validation set and to stop the algorithm when performance stopped improving.

Chapter 6. Implementation of the Detectors

In this chapter we describe how we implement our detectors. Our main concern is speed of execution. We would like detection to be as fast as possible. Our strategy is to re-use multi-resolution information wherever possible and to use a coarse-to-fine search strategy and various other heuristics to prune out unpromising object candidates.

6.1. Exhaustive Search

As explained in Chapter 2, each detector is specialized for a specific orientation, size, alignment, and intensity of the object. However, an object can occur at any position, size, orientation, and intensity in the image. Our approach is to use an exhaustive search along all these dimensions to find objects in the image. First, to be able to detect the object at any position in the image, we have to re-apply all the detectors at regularly spaced intervals in the image. At each of these sampling sites we evaluate the candidate at five different intensity corrections and select the one that gives the best response. Then, to detect the object at any size, we have to repeat this process for magnified and contracted versions of the original image. We search at scales of magnification that decrease by a multiplicative factor of $\sqrt[4]{2} = 1.189$. As we explain below, we chose an integer root of 2 so we could reuse information at each octave in this search through scale. We then combine the results of running all these detectors. If there are multiple detections at the same or adjacent locations and/or scales, the algorithm chooses the strongest detection.

Since it will be very time-consuming to evaluate the image in such an exhaustive fashion, we experimented with several methods for decreasing computation time, as we will describe later in this chapter.

6.2. Searching for Objects at One Size

In Figure 40 we outline the processing steps in searching for faces over one scaled version of the original image, i.e., faces of one size. First, we compute the wavelet transform of the entire image. We compute an overcomplete transform for each level of the wavelet transform. We indi-

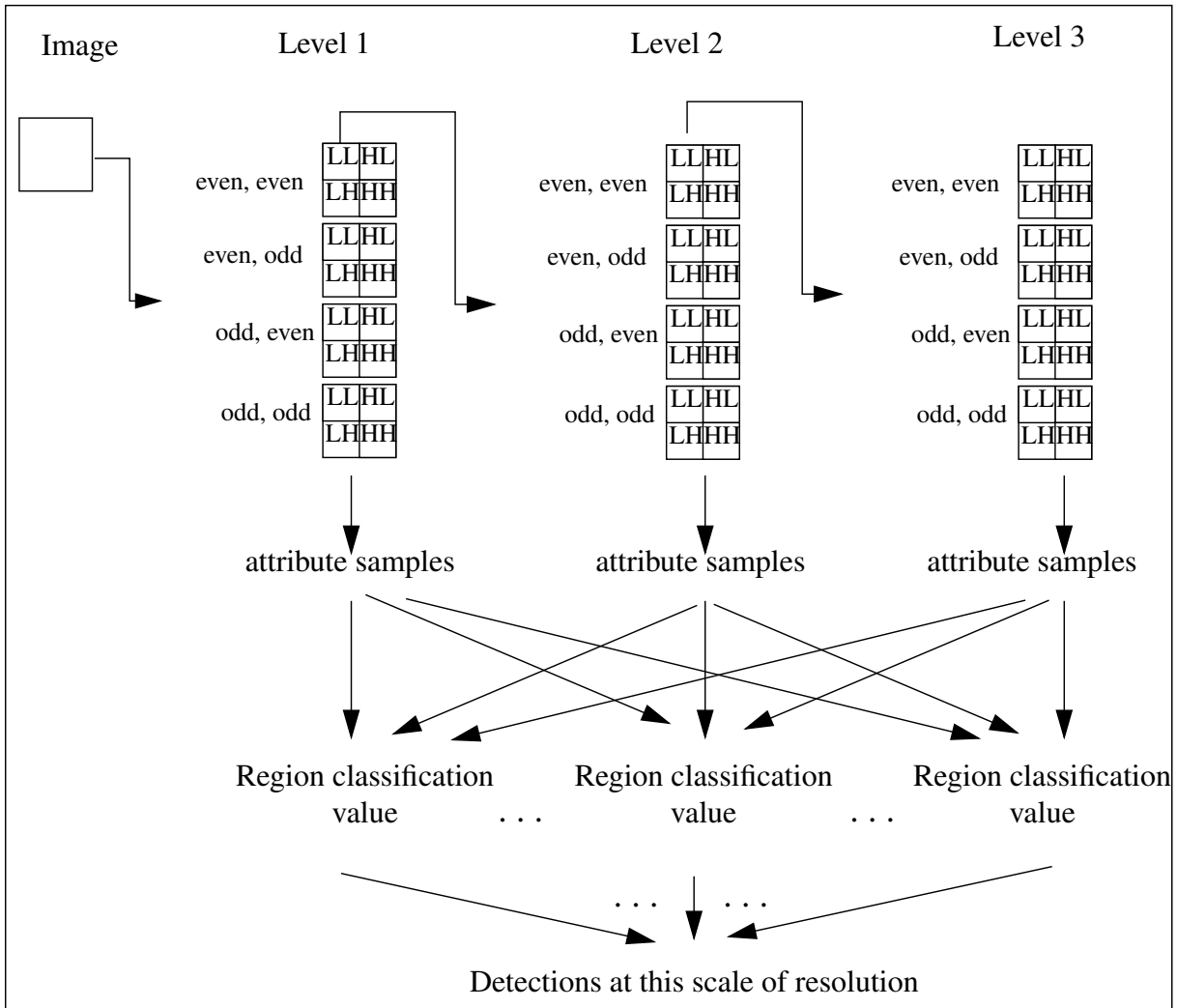


Figure 40. Sequence of operations for detection

cate the phases of the overcomplete transform as (even, even), (even, odd), (odd, even), and (odd, odd). We then compute attribute values from the wavelet coefficients at every location in the image. When then classify the image on a region by region basis, considering every possible overlapping candidate region. For each candidate region, we look up the probability of each attribute value. Each of these probabilities is a joint function of the attribute value and the position of the attribute sample within the candidate region. Rather than compute the output for each region using equation (15) in Chapter 3, we compute its logarithm. This transforms all the multiplications into additions:

$$\sum_{k=1}^{17} \sum_{x,\, y \in \text{region}} \log \frac{P_k(pattern_k(x, y), i(x), j(y) \mid \text{object})}{P_k(pattern_k(x, y), i(x), j(y) \mid \text{non-object})} \tag{40}$$

We then threshold on this value. Finally for each real face in the image there will be a cluster of candidate regions that all have a classification value over threshold. Below in Figure 41 we show the raw output from one image over 4 scales. (The value shown at each pixel corresponds to the output associated with a region centered at that position). Notice that the algorithm gives a high

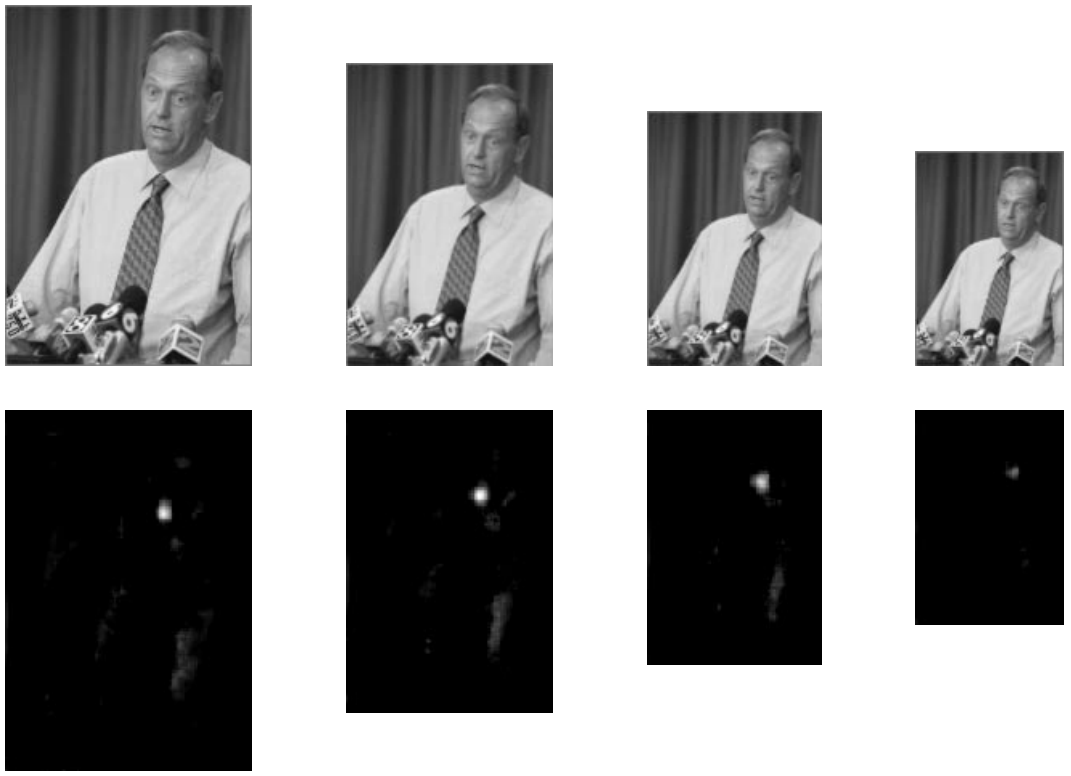


Figure 41. Region based response as a function of scale of magnification and position.

response to a cluster of candidate regions across position and scale. These clusters will usually correspond to one actual face in the image. We use the following strategy to merge detections that are nearby in position and scale. Once we have determined all the candidate regions that are above the detection threshold, we find the candidate region that gives the highest response. This candidate region is the first to be labelled as the object. We then remove from consideration all candidate regions that are within a radius in position and scale from this detection. For positional

radius we use one half the width of the object. For size we remove all candidates within this positional radius that vary from half to twice the detection's size. We then continue to search among the remaining candidate regions and find the one with next largest response and remove all candidates within the positional and scale radius of this detection. We continue this process until all candidate regions have been classified as an object or discarded.

The main computational bottleneck in detection is retrieving the probability values from the histograms. As mentioned above, we only have to compute the attribute values once for the entire image. However, for each sample site within each candidate object location we have to look up a probability indexed jointly by the attribute value and the position within the region. This way each sampling site is evaluated multiple times for its probability since it will be used as part of many candidate regions. Moreover, the tables we index into to retrieve these probabilities are quite large, with as many as 1.6 million entries. By working with data structures this large we are penalized in terms of memory performance. In particular, data structures this large will not fit into primary or secondary memory cache on most standard computers of today. When we make accesses into this data structure, the majority of them will be to main memory. Since main memory access can be as long as 100 clock cycles[80], these operations will lead to significant performance degradation.

6.3. Re-using Multi-resolution Information

In searching for the object, we use multi-resolution in two ways. First, we evaluate the input images over a number of scales of magnification to search for the object at different sizes. Then we evaluate each of these at three scales of resolution corresponding to the three levels in the wavelet transform. Rather than make all these evaluations separately, we can re-use many of them. Let us consider how this can be done. Let us consider the first four magnification scales at which we evaluate the image: 1.0, 1.189, 1.414 = 1.189^2, 1.681=1.189^3. For each of these we compute a three-level wavelet transform. We then compute our attribute values from the wavelet coefficients, and then look-up probabilities for the attribute values. We can then re-use much of this information when we evaluate the image at the 5th scale of magnification, 2.0. Currently we only re-use the level 3 LL subband as the input image. Conceivable we could also re-use the transform coefficients and the attribute values. The main obstacle to doing so is the amount of

memory that would be involved.

6.4. Heuristic Speed-ups

We experimented with several heuristics to speed up processing time. The most successful was a coarse-to-fine strategy to prune out unpromising candidates. We also experimented with color and adjacent heuristics.

6.4.1. Coarse to Fine Search Strategy

We divide the attributes into three sets: ones based on level 1 coefficients, ones based on level 2 coefficients, and ones based on level 3 coefficients. The first set we evaluate over the image at 1/16 of the normal resolution. We then threshold the partial output of the detector based on these attributes. We only continue searching those candidates whose probability is above this threshold. We repeat this process after we process the image using the level 2 attributes. The danger in this approach is that we may discard candidates that we would be correctly detected as faces if we fully evaluated them. For this reason we set the heuristic thresholds conservatively based on performance on a cross-evaluation set. Through this strategy we reduce our computational requirements by 1 to 2 orders of magnitude with little loss in accuracy.

6.4.2. Adjacent Heuristic

We also noted that we search at successive scales, that if we our response was low at one scale it will be low at the adjacent scale. We have experimented with thresholds on the value at the adjacent scale to prune out candidates at the current scale.

6.4.3. Color Heuristics

We also experimented with a color preprocessor to prune out unpromising candidates. We used an 8x8x8 histograms to represent the color distribution of skin-color and non-skin color. This also improved performance speed by a factor of 2 or 4 when combined with the other heuristic. It also helped discard a few candidates that would have otherwise been false detections. The main problem with color is that even with the color threshold set low, many actual faces get removed on images which have poor color balancing.

6.5. Performance Time

For faces we can evaluate a 240x256 image in about 90 seconds on average using a Pentium II at 450MHz.

For cars we can evaluate a 240x256 image in about 450 seconds on average using a Pentium II at 450MHz.

Chapter 7. Face Detection Performance

In this chapter we describe our results in face detection in Section 7.1, provide analysis of how the different parts of the face influence detection in Section 7.2, and assess statistical dependency across the extent of the face in Section 7.3.

7.1. Results in Face Detection

The distinguishing characteristic of our face detector is that it works for both frontal and out-of-plane rotational views. To date, several researchers [14][18][19][20][38] have had success developing algorithms that work for frontal views of faces, but none, to our knowledge, have had success with profile (side) views except [87] (below we will compare our performance with [87]).

We believe there are several reasons why profile view faces are more difficult to detect than frontal views. First, the salient features on the face (eyes, nose, and mouth) are not as prominent when viewed from the side as they are when they are viewed frontally. Also, for frontal views these features are interior to the object, whereas on a profile one of the strongest features is the silhouette with the background. Since the background can be almost any visual pattern, a profile detector must accommodate much more variation in the silhouette's appearance than a frontal detector does for interior features.

We compared the performance of our detectors with that reported by Rowley and Kanade [87] on a test set of profile views selected from a set of proprietary images Kodak provided to Carnegie Mellon University. These images consists of typical amateur photographs with some of the typical problems of such images, including poor lighting, contrast and focus. This test set consisted of 17 images with 46 faces, of which 36 are in profile view (between 3/4 view and full profile

view):

Table 8: Face Detection results on Kodak data set

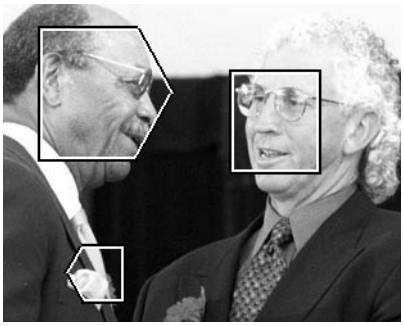
Rowley & Kanade [87]		Schneiderman and Kanade (using AdaBoost)			
Detection	False Detections	γ	Detection (all faces)	Detection (profiles only)	False Detections
58.7%	1347	0.5	80.4%	86.1%	105
41.3%	617	1.0	70.0%	69.4%	7
32.6%	136	1.5	63.0%	61.1%	1

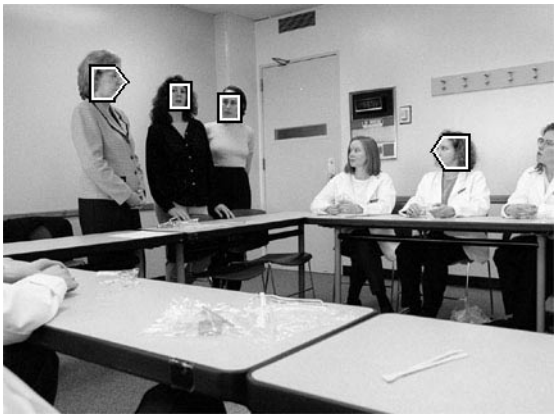
We also collected a larger test set for benchmarking face detection performance for out-of-plane rotation. This test set consists of 208 images with 441 faces that vary in pose from full frontal to side view. Of these images approximately 347 are profile view (between 3/4 view and full profile view). We gathered these images from a variety of sites on the World Wide Web, mainly news sites such as Yahoo and the New York Times. Most of these images were taken by professional photographers and of better quality than the Kodak images in terms of composition, contrast, and focus. Otherwise, they are unconstrained in terms of content, background scenery, and lighting. Below in Table 9 we show the performance at different values of the threshold γ controlling the sensitivity of the detectors. By changing γ we linearly scale the detection thresholds of both the profile and frontal detectors. We also compare the performance of the detectors trained with AdaBoost and without AdaBoost. Below in Figure 42 we show some typical results on this image set evaluated at $\gamma = 1.0$ using detectors trained with AdaBoost.

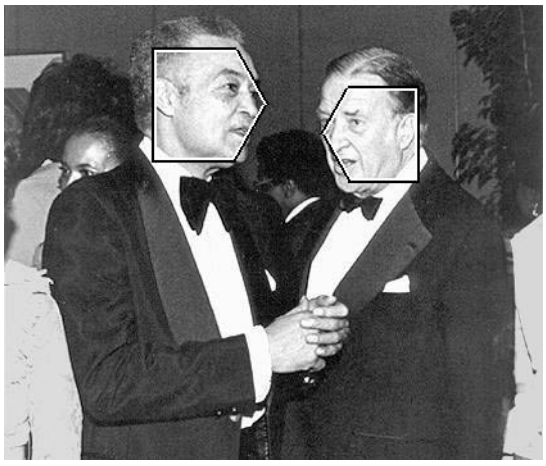
Table 9: Face Detection Results on Schneiderman & Kanade Test Set

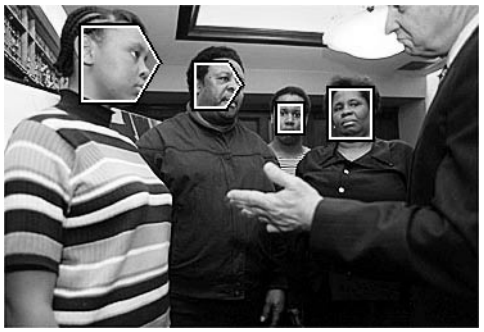
With AdaBoost				Without AdaBoost	
γ	Detections (all faces)	Detection (profiles)	False Detections	Detection (all faces)	False Detections
0.0	92.7%	92.8%	700	82%	137
1.5	85.5%	86.4%	91	74%	27
2.5	75.2%	78.6%	12	60%	3

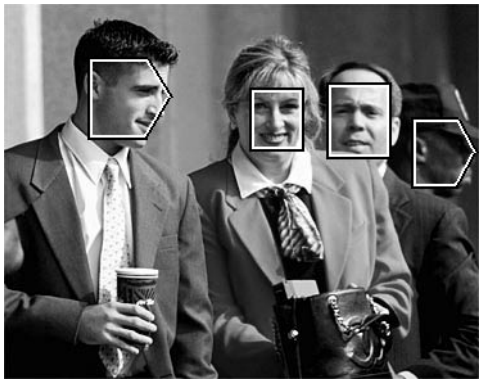
In terms of frontal face detection, the accuracy of our detector compares favorably with those

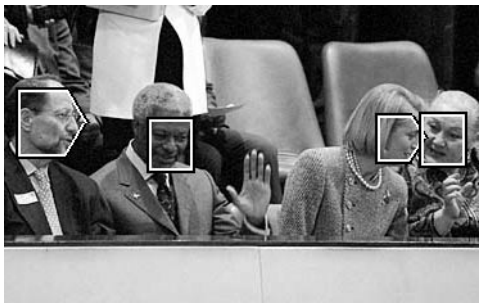












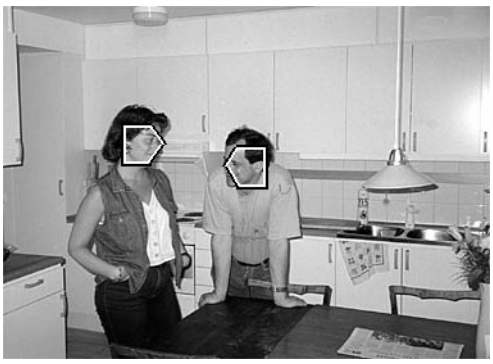


Figure 42. Face detection results

of other researchers. In these experiments, we also noticed some differences in performance between the detector described in this thesis and an improved version of the detector we described in [76]. Both of these detectors use similar probabilistic structures but differ mainly in that the detector in [76] uses visual attributes based on localized eigenvectors rather than wavelet coefficients. The wavelet based detector described in this thesis performs much better for profile view faces. However, the eigenvector based detector [76] seems to be perform slightly better on frontal faces. Below in Table 10 we compare our face detectors (wavelet-based and eigenvector-based [76]) with those results reported by others on the combined frontal face test set combining the test images from Sung and Poggio [18] and Rowley, Baluja, and Kanade [14].

Table 10: Frontal Face Detection on Sung & Poggio and Rowley [18] & Baluja & Kanade Combined Test Set [14]

	Detection Rate	False Detections
Schneiderman and Kanade* (wavelet)	94.4% (95.8%)	65
Roth, Yang, Ahuja *[38]	(94.8%)	78
Schneiderman and Kanade (eigenvector)* [76]	90.2% (91.8%)	110
Rowley, Baluja, and Kanade [14]	86.0%	31

* Indicates that 5 images of line drawn faces were excluded leaving 125 images with 483 labeled faces. However, there are at least 10 additional human faces that are not labeled in the ground truth for this test set. The numbers not in parentheses indicate results on just the 483 labeled faces. To be consistent with [38], we also indicate, in parentheses, the ratio between the total number of faces found by computer and 483 (the number labelled by hand).

7.2. Positional Decomposition of Face Detection

Since the detectors use all portions of the face, we wondered which regions tended to be most influential in detecting faces. For example, we wanted to know if the eyes, nose, and, mouth are really the most influential parts? To get a sense of how much the various parts of a face contributed to the overall detection score, we decomposed the contribution as a function of position on several faces; that is, for each face we plot an "influence image" that indicates the amount of influence at each position on the face. Brighter areas indicate strong positive influence, gray regions indicate neutral influence and darker areas indicate negative influence. As we would expect, areas that are occluded usually contributed a negative influence. In terms of positive influence, there did not seem to be a region that was consistently more influential than the others. In particular, the regions of positive influence were not sharply localized but tended to be spread out. In Figure 43 we show the corresponding "influence images" for several frontal faces and in Figure 44 we show the corresponding "influence images" for several profile faces.

7.3. Analysis of Pair-wise Statistical Dependency

We also evaluated pair-wise statistical dependency among wavelet coefficients for both frontal and right profile view. We evaluated pair-wise dependency using the following measure [64][85][86]:

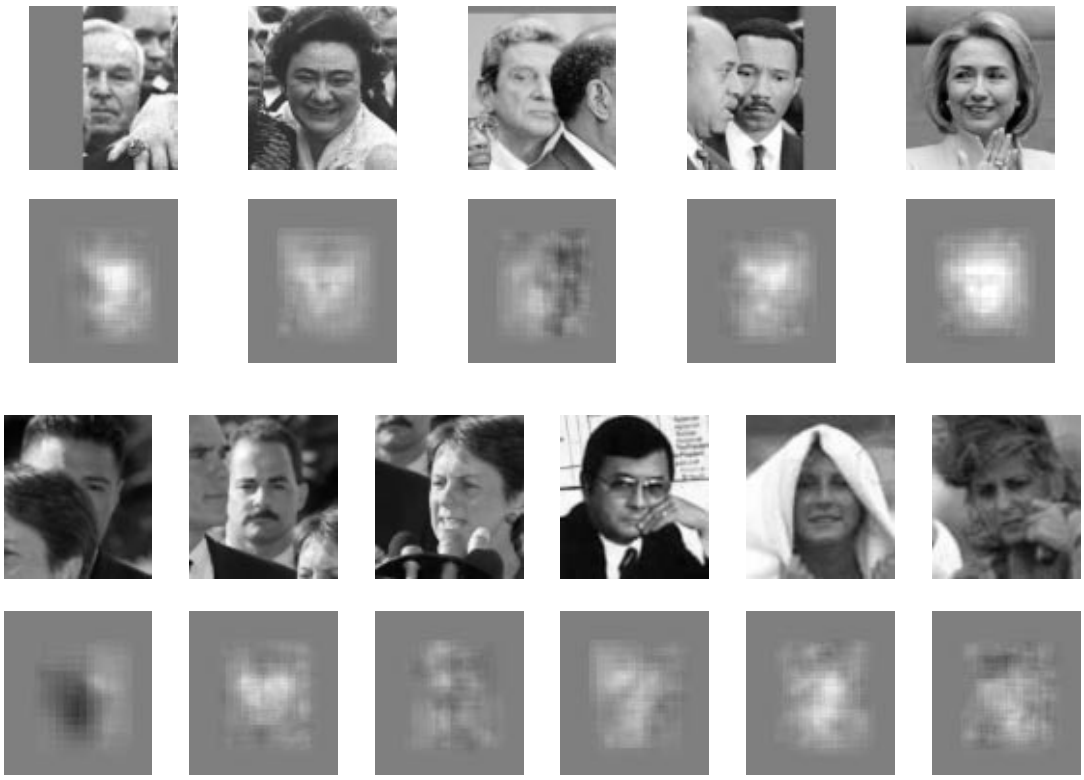


Figure 43. Frontal faces and their corresponding "influence images"

$$
\begin{aligned}
D(c_i, c_j) &= H(c_i) + H(c_j) - H(c_i, c_j) \\
H(c_i) &= \sum_{k=1}^{m} -P(c_i = v_k) \ln (P(c_i = v_k)) \\
H(c_i, c_j) &= \sum_{k=1}^{m} \sum_{l=1}^{m} -P(c_i = v_k, c_i = v_l) \ln P(c_i = v_k, c_i = v_l)
\end{aligned}
\tag{41}
$$

where c_i and c_j represent two wavelet coefficient that are discretized to take on m values, v_k.

Below we show some "dependency images" for individual coefficients. These images show the measure of pair-wise statistical dependency between a single coefficient and the rest of the coefficients. The statistical dependency is strongest between the coefficient and itself; thus the coefficient is indicated by the brightest spot in the wavelet transform. As we can see the extent of

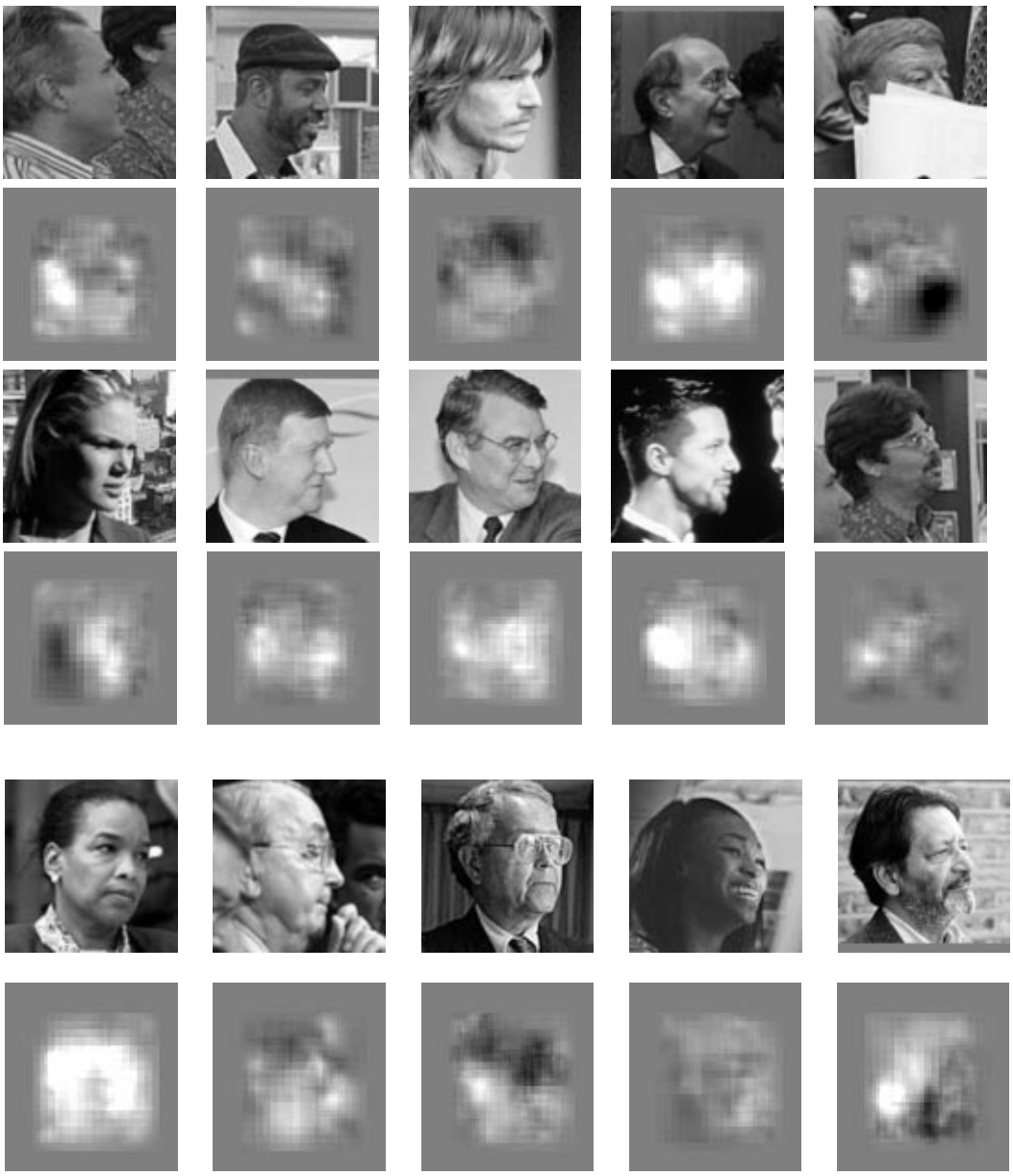


Figure 44. Profile images and their corresponding “influence” images

statistical dependency varies from coefficient to coefficient. The strongest dependencies tend to be inter-frequency over spatially registered positions. Also we notice that statistical dependency is not always localized and can exist over disjoint regions. In particular, we notice some statistical dependencies between the eye regions:

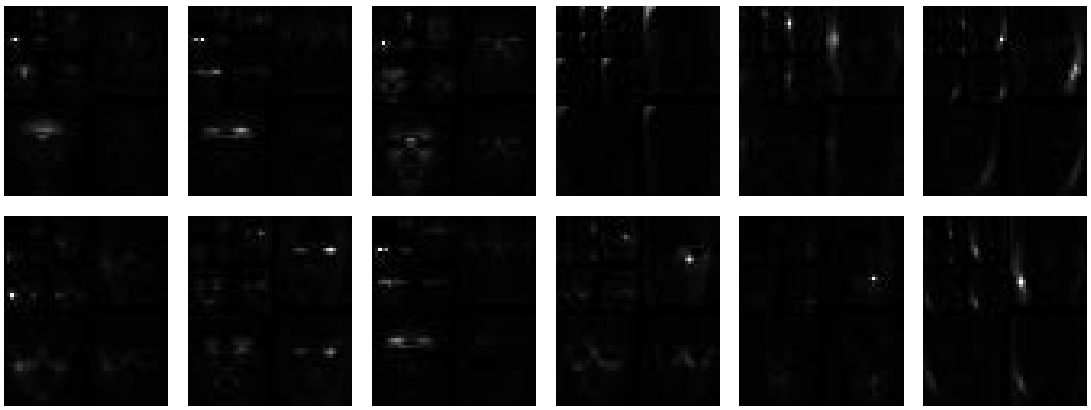


Figure 45. Statistical dependency for frontal faces. Each image gives a measure of the pair-wise statistical dependency between one coefficient and all the other coefficients in the wavelet transform.

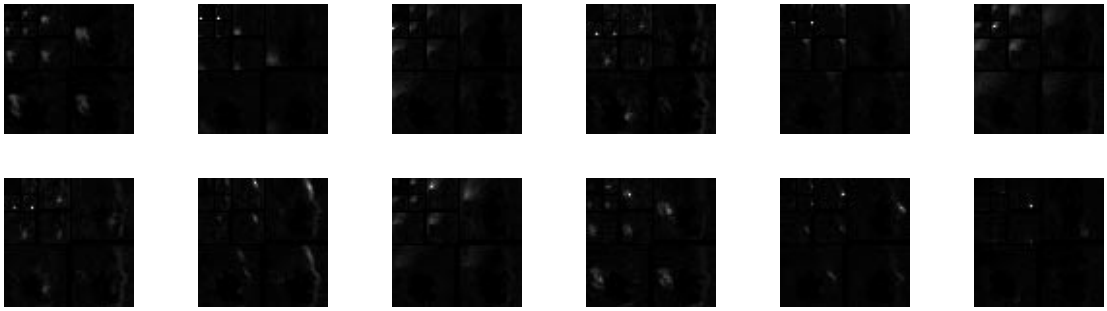


Figure 46. Statistical dependency for profile faces. Each image gives a measure of the pair-wise statistical dependency between one coefficient and all the other coefficients in the wavelet transform.

Figure 47. Statistical dependency for "non-object" images. Each image gives a measure of the pair-wise statistical dependency between one coefficient and all the other coefficients in the wavelet transform

Chapter 8. Car Detection Performance

In this chapter we describe our results in car detection in Section 8.1, provide analysis of how the different parts of the car influence detection in Section 8.2, and assess statistical dependency across the extent of the car in Section 8.3.

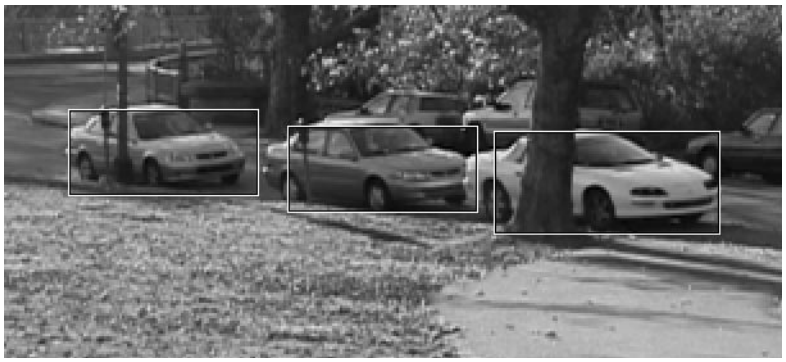
8.1. Results in Car Detection

To test the accuracy of the car detector, we collected a set of 104 images that contain 213 cars which span a wide variety of models, sizes, orientations, background scenery, lighting conditions and some partial occlusion. We gathered these images using several cameras and from sites on the World Wide Web. Overall our performance was as follows:

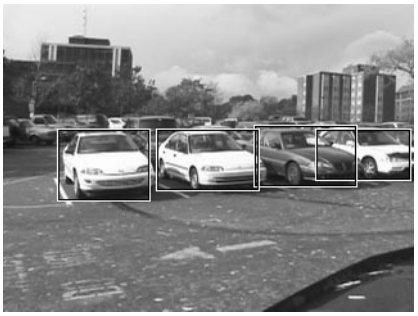
Table 11: Car Detection Results

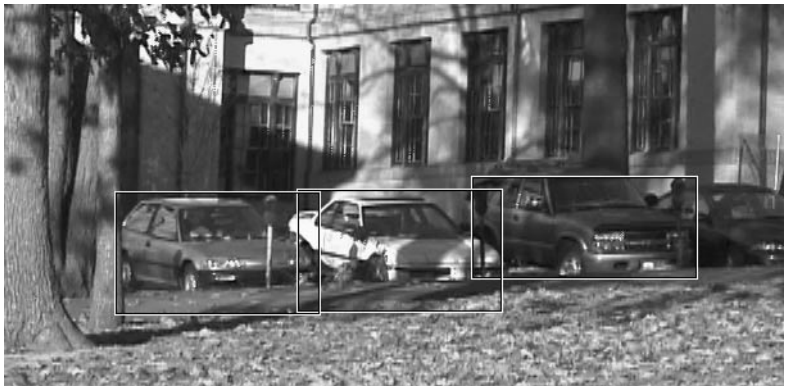
γ	Detections	Misses	False Detections
1.05	177 (83%)	36 (17%)	7
1.0	183 (86%)	30 (14%)	10
0.9	197 (92%)	16 (8%)	71

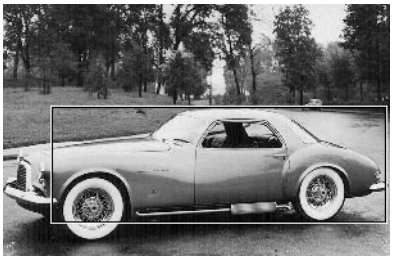
where γ controls the sensitivity of the detectors; that is, by scaling γ we linearly scale the detection thresholds of all eight car detectors. Below we show some typical results on this image set evaluated at $\gamma = 1.0$:




www.corbis.com








Factory Style with light - $115
Maxima 87 - 88
Aero Kit - $535
Eclipse/Laser/Talon
90 - 94 Kit - $635
200SX 95 - on
Factory Style with Light - $115
Over 40 factory and custom style wings to choose from. New applications added weekly.
Over 120 different ground effects! Lifetime Warranty.
Accord 92- 93
4D Kit - $449

8.2. Positional Decomposition of Car Detection

Since the detectors use all portions of the car, we wondered which regions tended to be most influential. To get a sense of how much the various parts of a car contributed to the overall detection score, we decomposed the contribution as a function of position on several car; that is, for each car we plot an "influence image" that indicates the amount of influence at each position on the car. Brighter areas indicate strong positive influence, gray regions indicate neutral influence and darker areas indicate negative influence Below in Figure 48 we give some examples.

Figure 48. "Influence images" for various car inputs

8.3. Analysis of Pair-wise Statistical Dependency

We also evaluated pair-wise statistical dependency among wavelet coefficients for both frontal and right profile view. We evaluated pair-wise dependency using the following measure [64][85][86]:

$$
\begin{aligned}
D(c_i, c_j) &= H(c_i) + H(c_j) - H(c_i, c_j) \\
H(c_i) &= \sum_{k=1}^{m} -P(c_i = v_k)\ln(P(c_i = v_k)) \\
H(c_i, c_j) &= \sum_{k=1}^{m}\sum_{l=1}^{m} -P(c_i = v_k, c_i = v_l)\ln P(c_i = v_k, c_i = v_l)
\end{aligned}
\tag{42}
$$

where c_i and c_j represent two wavelet coefficient that are discretized to take on m values, v_k.

Below we show some "dependency images" for individual coefficients. These images show the measure of statistical dependency between a single coefficient and the rest of the coefficients. The statistical dependency is strongest between the coefficient and itself; thus the coefficient is indicated by the brightest spot in the wavelet transform. As we can see the extent of statistical dependency varies from coefficient to coefficient. The strongest dependencies tend to be inter-frequency over spatially registered positions. Also we notice that statistical dependency is not always localized and can exist over disjoint regions. In Figures 49 - 51 we give some examples.

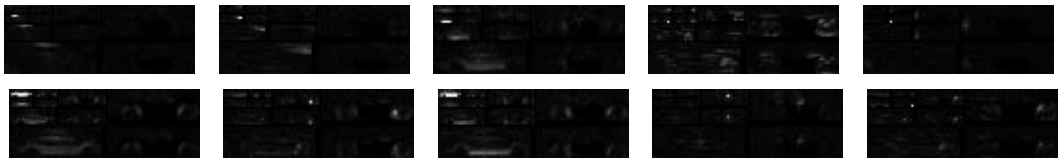


Figure 49. Statistical dependency for car (car viewpoint #8). Each image gives a measure of the pair-wise statistical dependency between one coefficient and all the other coefficients in the transform

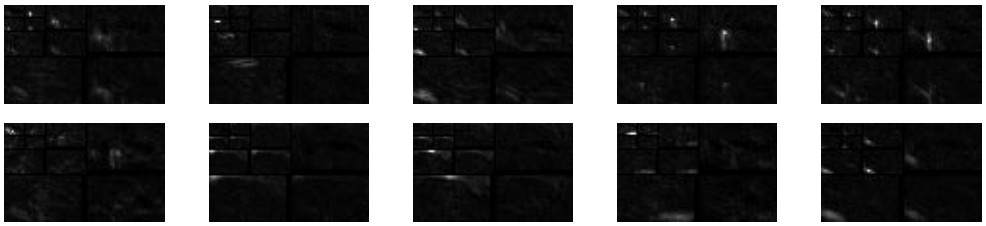


Figure 50. Statistical dependency for car (car viewpoint #5). Each image gives a measure of the pair-wise statistical dependency between one coefficient and all the other coefficients in the transform

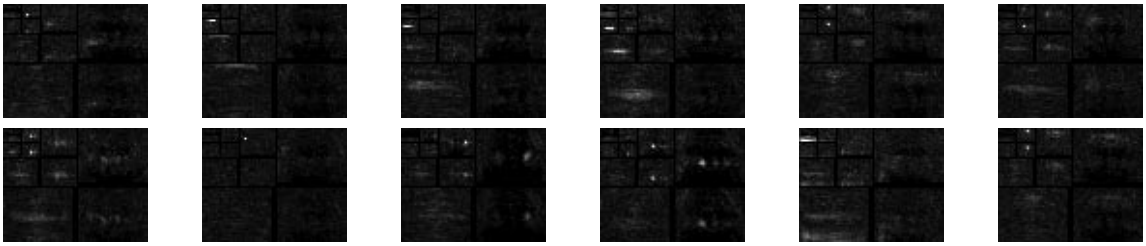


Figure 51.Statistical dependency for car (car viewpoint #1). Each image gives a measure of the pair-wise statistical dependency between one coefficient and all the other coefficients in the transform

Chapter 9. Review of Other Statistical Detection Methods

In this chapter in Section 9.1 we describe some of the major theoretical differences between our method of object detection and other methods of object detection and in Sections 9.2 and 9.3 we summarize several methods that have been applied to face and car detection. In this discussion we emphasize the particular modeling choices in each of these methods.

9.1. Comparison of Our Approach to Previous Detection / Recognition Methods

Below we summarize the main difference between our approach and other previous approaches to object detection / recognition

9.1.1. Local Appearance Versus Global Appearance

Much work in object recognition treats the appearance of the object in terms of full-sized rigid templates including the work of [5],[6], [30], [44], [29]. These methods represent the appearance of the entire object as one entity rather than decomposing the object into smaller parts. There are several disadvantages to this type of model. First, the global methods that involve dimensionality reduction [5],[6], [30] will end up emphasizing the coarse attributes of object appearance rather than the distinctive nature of the smaller parts such as the eyes, nose, and mouth on a face. Second the matching of large template is known to be sensitive to small differences in scale, position, and orientation. Finally the matching of large regions can also be strongly influenced by "irrelevant" pixels. On many objects, such as a car, there will be large indistinctive areas such as the hood and windshield that are punctuated by relatively smaller areas of distinctive detailing such as the grill and headlights. In matching a large region, the majority of the pixels will come from the untextured parts and dominate selection of the match (using any norm that weighs each pixel equally such as L1 or L2).

9.1.2. Sampling of Local Appearance

Most methods that use an local appearance representation [26], [4], [40], [78], [79] use a mini-

mal set of hand-picked features, such as the eyes, nose, mouth on a face or a minimal set determined by an automatic method [39], [81], [82], [83]. In contrast, methods such as [9][20] sample the object everywhere. We take this a step further by sampling with overlap and sampling multiple attributes at each location. Our strategy in doing so is use as much information as possible in making our detection decision. As with any type of decision problem, by using more information we improve our probability of making correct detections. For example, in face detection we can rule out many candidate faces on the basis of inappropriate texture or features in areas we expect a nearly uniform texture such as on the cheeks of a human face.

9.1.3. Representation of Geometric Relationships

As we mentioned earlier, an important cue for detection is the geometric relationships of the various local appearance patches. Some approaches do not model the geometry among local areas [84][9]. Other methods represent geometry rigidly [14], [19], [81], [82], [83], [38] and may be brittle to small variation in the part positions. Other methods use graph based matching techniques where nodes represent features and edges represent distances between feature pairs [78][79]. With this approach computational complexity grows polynomially with the number of features. Others depend on a coordinate system defined by two or three features [26][32] and may be sensitive to any errors in these feature positions. Our approach using multidimensional histograms allows for flexibility in feature position with respect to a stable coordinate system and linear cost in the number of features and positional resolution. We allow for flexibility in feature position by representing position, x, y, at a coarse resolution in the histogram, P(*pattern*, x, y) or we can smooth the positional component of the histogram by convolving it with a Gaussian in x,y as we did in [76]. Also, by using a histogram we can process the image in a fixed repetitive fashion. In contrast, methods that use a set of hand-picked features step through the image in irregular increments and incur additional computational cost by doing so [80].

9.1.4. Representation of Local Appearance using Wavelets

Haar wavelets [81], [82], [83], Gaussian pyramid representation [84] and Gabor wavelets [78], [79] are all multi-resolution based representations that have been used in object recognition/detection. Haar wavelets are an orthogonal basis that achieves a piece-wise constant approximation to the image. We use 5/3 linear phase wavelet based representation which may have some advan-

tages of the Haar wavelets in terms of smoothness and sparseness. Also unlike the Gaussian pyramid representation [84] and Gabor wavelets [78], [79] our wavelet representation has no redundancy; that is the transform is the same size as the original image. Also, our representation is unique in decomposing appearance along the dimensions of space, frequency, and orientation.

9.1.5. Representation of Statistics/Discriminant Functions

Research in object recognition has spanned a number of discriminant functions statistical models. [18] and [14] use artificial neural networks (multilayer perceptrons) as part of their discriminant functions. This discriminant function is attractive in that it can achieve complicated boundaries in input space. There are several disadvantages also. The method for estimating such a model from data does not have a closed form solution. All solution methods involve iterative search and the most popular method (backpropagation) is simply gradient descent. These methods can become trapped in local minima. Mixture models [18][40] are also attractive because they can achieve complicated boundaries in input space but are also susceptible to local minima. Quadratic discriminant functions [19]. [81], [82], [83] achieve partition of input space into conic sections.

In previous work in computer vision, histograms have only be used for representing the statistics of appearance. [54] uses a histogram to represent color statistics and [9] uses a histogram to compute statistics of Gaussian derivatives. One of the main contributions of this research is to use histograms to jointly represent the statistics of visual attributes and position on the object as originally published in [76]. Others have since used a similar representation [10].

9.1.6. Estimation of Statistics

Since classification accuracy is more important than accuracy of probabilistic models that form the classifier, better performance can be achieved by training the detector to minimize classification error. To do so, we use the AdaBoost algorithm [70], [72] to explicitly reduce classification error. This algorithm works under a very similar principle to support vector machines [15] as used by [19] [81], [82], [83] by giving more weight to the training examples that are closet to the decision boundary between the two classes.

9.2. Summary of Previous Methods for Face Detection

9.2.1. Sung and Poggio [18]

Sung and Poggio developed one of the first methods for frontal face detection. They use a 19x19 region as their standard size input region. They normalize the intensities within this region by first subtracting the best fit intensity plane and then performing histogram normalization.

Their model consists of two parts: a mixture of Gaussians model and a multilayer perceptron network. In the Gaussian part of the model, the face and non-face classes are each represented as a mixture of 6 Gaussians. In the mixture formulation, each Gaussian receives equal weighting. The centers, μ_i, and covariances, Σ_i, of these Gaussians are estimated using a modified k-means algorithm on labeled training samples.

This representation could be viewed as forming class-conditional distributions are given by the mixtures

$$\begin{aligned} P(region|\text{object}) &\approx \sum_{i=1}^{6} \frac{1}{\sqrt{(2\pi)^n|\Sigma|}} \exp((-0.5)(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)) \\ P(region|\overline{\text{object}}) &\approx \sum_{i=1}^{6} \frac{1}{\sqrt{(2\pi)^n|\Sigma`|}} \exp((-0.5)(x-\mu_i`)^T\Sigma_i^{-1}(x-\mu_i`)) \end{aligned} \tag{43}$$

However, instead of using these for classification (as in a likelihood ratio test), they input each of the Mahalanobis distances,

$$d = (x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)$$

into a multilayer perceptron which generates an output from 0 to 1. They approximate the Mahalanobis distance by computing the true Mahalanobis distance along the 75 principal dimensions of the space and adding it to the Euclidean distance along the remaining dimensions (remember the entire input space is 19x19).

The addition of the multilayer perceptron neural network makes the algorithm difficult to ana-

lyze. The multilayer perceptron does not increase the representational power over equation (43) since the Mahalanobis distances are a sufficient statistic. The main advantage would be that by training the neural network, they would be minimizing classification error rather than basing their detector on probability estimating.

9.2.2. Rowley, Baluja, and Kanade [14]

Soon after Sung and Poggio's algorithm, Rowley, Baluja, and Kanade developed a slightly more accurate method for frontal face detection. The method of Rowley, Baluja, and Kanade uses a 20x20 input region that is presented to a multilayer perceptron neural network system for classification. They have performed experiments with both single neural networks and modular systems consisting of several neural networks.

In all their algorithms, each 20x20 input region is pre-processed to correct for differences in lighting conditions using the same method as Sung and Poggio [18]: a linear function of intensity is fit to the region and subtracted out, then histogram normalization is performed.

The neural network topology they used in all their experiments consisted of one layer of hidden units where each hidden has a receptive field of either 5x5, 10x10, or 20x5. The used two variations on this topology: one with 52 hidden units and the other with 78 hidden units.

In their experiments with modular systems, they separately trained two or three of the above networks and then applied various methods for merging their results, including training another multilayer perceptron or an individual perceptron to act as an arbitrator.

They experimented with various heuristics to the output of the neural network, including merging nearby detections, discarding isolated detections, and not allowing overlapping faces.

Its somewhat difficult to analyze the behavior of a multilayer perceptron. We can infer that this particular network topology emphasizes local features over global ones, since the hidden units have only local support. Statistical dependencies that span a larger region than any of the receptive fields are only modeled weakly through normalization and network arbitration.

9.2.3. Osuna, Freund, and Girosi [19]

Osuna, Freund, and Girosi use a 19x19 input region, x, for detecting faces. They apply the following discriminant function to the input image, x (see [24] or [25] for definition of "2nd-degree polynomial kernel function"):

$$f(x) = \sum_{i \in SV} c_i (y_i^T x + 1)^2 \tag{44}$$

Where *SV*, indicates support vector.

This function can be rewritten as a quadratic discriminant function:

$$f(x) = a_0 + a_1^T x + x^T A_2 x \tag{45}$$

where a_0 is a scalar, a_1 is a vector, and A_2 is a symmetric semi-positive definite matrix. These parameters can be directly computed from c_i, y_i, and c_0 given above.

The quantities, c_i and y_i are determined through training. Their method of training is explicitly designed to minimize generalization error.

One way we can compare this model to the general model of the posterior probability function is through the concept of Taylor approximation. We can think of a quadratic discriminant function loosely as being a Taylor series approximation to the posterior probability function truncated at its 2nd order term. Likewise, we can think of a perceptron as being a Taylor series approximation to the posterior probability function truncated at first order.

9.2.4. Moghaddam and Pentland [12]

Moghaddam and Pentland have developed a face detection algorithm as part of a bigger system that includes face recognition and face tracking. For face detection, they appear to use a standard sized input region. They normalize the intensity of the region by scaling the image to have zero mean and unit variance.

Moghaddam and Pentland base their method of detection on the value of the class conditional probability function, $P(region|object)$. Their model of the class conditional density is decomposed into two independent distributions:

$$p(region = X|object) \approx P_1(F^T X|object)P_2((I - F^T)X|object)$$

The first distribution, $P_1(F^T X|object)$, is computed on a projection of the image region onto a 10 dimensional subspace, given by the columns of *F*. The subspace, *F*, consists of the principal components of the face data set. They model this distribution as a mixture of Gaussians:

$$P_1(F^T X|object) \approx \sum_i \pi_i N(\mu_i, \Sigma_i)$$

where

$$1 = \sum_i \pi_i$$

All the parameters in this distribution -- the weighting coefficients, the means, the covariances are estimated using the maximum likelihood principle and solved using the E-M algorithm, which finds a locally optimal solution.

The 2nd distribution, $P_2((I - F^T)X|object)$, computes the distributions of the projection of the image region onto the space orthogonal to the 10 principal components. This distribution is modeled as a Gaussian with zero mean and scaled identity matrix for the covariance:

$$P_2((I - F^T)X|object) \approx N(0, I\hat{\sigma}^2)$$

where

$$\hat{\sigma}^2 = \frac{1}{n-10}\sum_{i=11}^{n}\lambda_i$$

where the λ_i's are the eigenvalues of the distribution.

9.2.5. Colmenarez and Huang [20]

Colmenarez and Huang have developed a method for face detection. They use a Markov field model to capture the statistical behavior of the face and non-faces classes over a 11x11 input region. Their decision rule is based on thresholding the likelihood ratio:

$$L = \frac{P(region|object)}{P(region|\overline{object})} \tag{46}$$

They model both the face and non-face class-conditional probabilities using a first order Markov field model over the 11x11 input region. The relationship between this Markov field model and the general form of the class-conditional probability function, $P(region|object)$ is fairly straightforward. Consider re-ordering the image *region* into a 1 dimensional array of *n* pixels. The general expression for class conditional probability of the image *region* can then be re-written using the chain rule:

$$
\begin{aligned}
&P(region|object) = \\
&P(region(n), region(n-1), \ldots, region(1)|object) = \\
&\qquad P(region(n)|region(n-1), \ldots, region(1), object) \text{ x} \\
&\qquad P(region(n-1)|region(n-2), \ldots, region(1), object) \text{ x} \\
&\qquad . \\
&\qquad . \\
&\qquad P(region(1)|object)
\end{aligned}
\tag{47}
$$

where *region(i)* is the pixel intensity for the i^{th} pixel in the *region* and case $n = 121$ for Colmenarez and Huang's representation.

In applying the Markov assumption, it is assumed that a pixel is only statistically dependent on the k pixels that immediately precede it in the ordering of the 1-dimensional array:

$$
\begin{aligned}
&P(image(j)|image(j-1), image(j-2), \ldots, image(1), object) \approx \\
&\qquad P(image(j)|image(j-1), image(j-2), \ldots, image(j-k), object)
\end{aligned}
\tag{48}
$$

Colmenarez and Huang assume a 1st order Markov process, $k = 1$:

$$
\begin{aligned}
&P(region(j)|region(j-1), region(j-2), \ldots, region(1), object) \approx \\
&\qquad P(region(j)|region(j-1), object)
\end{aligned}
\tag{49}
$$

The class conditional density is then approximated by making this substitution for all terms on the right hand side of equation (47):

$$P(region(n), region(n-1), \ldots, region(1)|object) \approx$$

$$P(region(n)|region(n-1), object) \text{ x} \tag{50}$$

$$P(region(n-1)|region(n-2), object) \ldots P(region(1)|object)$$

Colmenarez and Huang consider all possible orderings of the pixels in mapping the 11x11 region into a 1 dimensional array. They select the mapping that optimizes the Kullback divergence of the face class with respect to the non-face class. To compute the Kullback divergence, training examples are used. They do not state why they use the Kullback divergence.

In building the overall model, they construct intermediate models for the 1st order conditional probabilities:

$$P(image(j)|image(j-1), object) = \frac{P(image(j), image(j-1)|object)}{P(image(j-1)|object)} \tag{51}$$

for all pixels pairs. They directly estimate the two terms on the right hand side of the equation. Each of these terms is a discrete valued function where each intensity value is quantized into 3 levels.

9.2.6. Burl and Perona [26]

Burl and Perona have developed a method for face detection. Burl and Perona's method emphasizes the spatial distribution of features. They detect 5 types of features on the face: the left eye, right eye, nose/lip junction, left nostril, and right nostril. They assume that the feature detectors for each feature are fallible. They may not respond to the actual feature, and they may give false responses elsewhere. Since they assume only one face is present in each image, they assume that at most one feature response is correct for each type of detector and all other responses to that detector are false alarms.

To locate each face, they consider all possible "hypotheses". In each hypothesis, H_i, only one (or none) of the responses for each detector is selected. Of all these hypotheses, they select the

one that gives the maximum posterior probability:

$$H^* = \arg max P(H_i|W) = \arg max\left(\frac{P(W|H_i)Pr(H_i)}{P(W)}\right) \quad (52)$$

where *W* represents the locations for all feature responses for all detectors.

Since *P(W)* is the same for all hypotheses, they maximize a "goodness" function:

$$G(H_i) = P(W|H_i)Pr(H_i) \quad (53)$$

Their prior probability, $Pr(H_i)$ depends only on how many (and which) of the 5 features are in the hypothesis. $Pr(H_i)$ will differ between two hypotheses only if one hypothesis includes a feature type (e.g. eye) that is absent in the other hypothesis. The difference depends primarily on a parameter, γ_j, which represents prior knowledge about the accuracy of each feature detector, *j*.

To model *P(W|H)* for each hypothesis, the data is partitioned into 2 sets: *W(H)* - the features responses hypothesized to correspond to the true features and *W(H')* - the remaining feature responses (false alarms). These two sets of feature responses are modeled by two statistically independent distributions:

$$P(W|H) = P_H(W(H))Q_{H'}(W(H')) \quad (54)$$

They further assume that the false alarms are statistically independent:

$$Q_{H'}(W(H')) = \prod q(W(H')) \quad (55)$$

where *q(w)* is the probability of an individual false alarm.

To simplify the goodness functions, it is divided by a constant given by Π q(w):

$$G_o(H_i) = \frac{P(W|H_i)Pr(H_i)}{\prod q(w)} \tag{56}$$

By substituting (55) into (54) and substituting the result into (56), each goodness function becomes a function of only the feature responses in the hypothesis, W(H):

$$G_o(H_i) = \frac{P(W|H_i)Pr(H_i)}{\prod q(W(H))} \tag{57}$$

Berl and Perona attempt to compensate for differences in translation, scale and rotation of the feature positions by defining a normalized coordinate system for each hypothesis. The location of the response of the first feature, the left eye, becomes the origin of the new coordinate system and the 2nd feature point, the right eye, is defined to be (1, 0). The coordinates of all other features can then be computed with respect to these two points. To allow for the possibility of missing features, they also compute coordinate systems defined by other pairs of feature points. Although this method removes the differences in translation, scale, and rotation with respect to the detected features, it is not stated how the feature detectors themselves are made invariant to scale and rotation. In making this normalization, the goodness function becomes:

$$G_o(H_i) = \frac{R(S)Pr(H_i)}{Q(S)} \tag{58}$$

where S describes the coordinates of the features in the normalized coordinate frame.

In this form, equation (58), we can draw some comparisons to our method of representation. We can think of the goodness function as being proportional to the posterior probability of the object conditioned on the normalized feature locations, S:

$$G_o(H_i) \propto \frac{P(S|object)P(object)}{P(S)} \tag{59}$$

The class conditional distribution of the features given the object, P(S|object), can be re-written decomposing position from appearance using the probability chain rule:

$$P(object|S) = P(s_1, w_1, \ldots, s_5, w_5|object) = P(s_1, \ldots, s_5|w_1, \ldots, w_5, object)P(w_1, \ldots, w_5|object) \quad (60)$$

The first term on the right of (60) embodies their description of shape and is modeled by a Gaussian:

$$P(s_1, \ldots, s_5|w_1, \ldots, w_5, object) = R(S) \approx N(\mu, \Sigma) \quad (61)$$

where μ and Σ are estimated from the training data.

The second probability on the right of (60) gives the probability of locating the actual features. This is given by their notion of prior probability Pr(H) and is not computed from the training data:

$$P(w_1, \ldots, w_5|object) \approx Pr(H_i) \quad (62)$$

The term in the denominator of (58) represents the unconditional distribution of the features. They represent the features as independently distributed with Gaussian distribution centered at the center of the image with variance σ^2 (no training data was used to estimate this distribution):

$$P(S) = Q(S) \approx \prod N(0, \sigma^2) \quad (63)$$

They report performances 80% and 94% recognition on various test sets they compiled. And they report 63% recognition on their training set.

9.2.7. Roth, Yang, Ahuja [38]

This work describes a method for frontal face detection on 20x20 regions. For every possible pixel value at every possible location within this region they assign a weight, giving a total of 20x20x256 weights. They use an iterative training procedure using the Winnow update rule to determine these weights. Once they have determined the weights they can classify any region by looking up and summing the weights corresponding to each pixel value:

(64)

$$\sum_{x,\, y \in\ 20x20} w(I(x, y), x, y)$$

If this sum exceeds a threshold the region is classified as a face.

This method performs quite well on the Sung & Poggio and Rowley, Baluja, Kanade Test set. They report 94.8% detection with 78 false detections. It is somewhat surprising that this method works so accurately since the features essentially have no spatial extent. They are just individual pixels.

9.3. Summary of Previous Methods for Car Detection

9.3.1. Rajagopalan, Burlina, Chellappa [44]

This paper describes a method for car detection from aerial images. They use a distance based classification metric on 16x16 regions. They cluster their training images into several classes of cars and several classes of non-cars. For each 16x16 input region, they compute the distance to each class. If the input is closest to a car cluster and under some threshold they classify it as a car. The distance threshold they use could be thought of as a Mahalanobis-like distance metric, except instead of normalizing distance by just 2nd order statistical moments, as in Mahalanobis distance, they use some higher order moments also. They have reported some success in detecting cars from this vantage.

9.3.2. Papageorgiou, Poggio [83]

In this method the Haar wavelet transform is taken of each input region. The wavelet coefficients from two of the middle frequency bands (3,030) wavelet coefficients are used as input to a quadratic classifier. The coefficients in the quadratic classifier are learned by using the Support Vector Machine training method. They report some success in detecting straight-on frontal and straight-on rear views.

Chapter 10. Conclusion

In this thesis we have advanced the state of the art in 3D object detection in the following ways. We have developed the first algorithm that can reliably detect faces that vary in viewpoint from frontal to side view. Previously only frontal face detection had been demonstrated reliably. We have also demonstrated the first method for car detection that works robustly over a range of view points.

Several concepts contribute to the effectiveness of these methods:

• Joint statistics of appearance and position - Much research in "parts-based" approaches to object recognition overlook the importance of representing the geometric arrangement of the parts. In our experiments, we have found performance improves drastically when we model the statistics of appearance and position jointly.

• Powerful representation of appearance - In our experiments we have observed that increased representation power improves the accuracy of the object detector. For example, we originally developed a weaker representation based on a subset of localized eigenvectors [76]. Although this representation worked well for frontal face detection, it was not fully satisfactory for profile detection. We also noticed that when we reconstructed profile images from this representation, many of the small features that form of the silhouette of the face were lost. We then redesigned our representation using the wavelet-based representation described in this thesis. With this new representation, our visual representation of these features was better leading to improved detection performance for profile views of faces.

• Representation of the non-object - We also observed that performance depended on how we represented the non-object class. We noticed that having some model was an improvement over having no model, even if our model was based on randomly sampled non-object images. Performance improved further by using bootstrapping to select non-object samples and improved still further by using AdaBoost to weight them.

• Visual cues based on local relationships - We have shown that by using a combination of visual cues with selective localization in space, frequency and orientation we can achieve accurate detection of faces and cars.

• Coarse-to-fine heuristics - By using coarse to fine heuristics, we have demonstrated that we can use a large model with many visual cues in a computationally feasible way.

There are several research areas we see as a natural continuation of this work:

• Representation - Representation remains the most important issue in object detection. Perhaps, some day researchers will develop specific statistic models for visual appearance in the same way Gaussian and Poisson models were developed to model specific physical phenomena. Of course, to do so, we would need a way to cope with the high dimensional nature of images. One approach would be to look at the pair-wise statistics among wavelet coefficients as we have suggested in Chapters 7 and 8. Another approach would be to use our knowledge of the physics of image formation. Good models for the effects of illumination, reflection, geometry, and material type on appearance exist. These models have been used to synthesize images that look fairly realistic. It may be possible to use such models to characterize the statistics of appearance. In particular, it may be easier to characterize the statistical variation of the "input" to the image formation process -- the illumination, the geometry of the scene, and the surface characteristics -- than to characterize the image variation directly. We could then analyze how these imaging models transform these stochastic inputs and thereby indirectly arrive at a statistical characterization of appearance.

• Intensity/Lighting correction - In our work, performance improved when we were able to correct for differences in illumination. Most of the existing methods for intensity correction use simple methods such as histogram equalization or transforming the intensities to have zero mean and unit variance. We believe that better performance can be achieved by explicitly accounting for the appearance of the object we are trying to enhance. One approach would be to use a probabilistic model of the object's appearance to choose the correction. Let us assume we have some lighting correction model:

$$image' = C(image, \theta) \tag{65}$$

where the parameter, θ, controls the lighting correction. Given that we have models for *P*(*image* | object) and *P*(*image* | non-object), we could choose the value of θ that gives the highest response and therefore is corrected so it most looks like a member of its class:

$$\hat{\theta} = \arg max_{\theta}\left(\frac{P(C(image, \theta)|\text{object})}{P(C(image, \theta)|\text{non-object})}\right) \tag{66}$$

We have conducted some preliminary experimentation with this approach but have not had much success.

• Sample selection and weighting - In our experiments, performance improved when we used bootstrapping to select samples and boosting to weight samples. Perhaps there is a principled way of combining these two methods to achieve better performance.

• Coding of appearance - We have used scalar quantization to discretize each wavelet coefficient separately. Methods of vector quantization whereby a whole group of coefficients is quantized together may improve performance. We noticed such an improvement in an earlier method based on eigenvector responses [76].

There are also several research problems that we see as a natural continuation of this work:

• Detection of other rigid objects - We would like to test the generality of this algorithm by applying it to other rigid objects such as boats, airplanes, animals, pedestrians, etc.

• Detection of more challenging objects - There are more challenging objects we would like to detect such as buildings, trees, and text in video. These objects have some structural regularity but less so than faces or cars. We believe it is possible to detect such objects accurately with current computing power, but new representations will have to be developed to do so.

• Other classifications - There are many other classification problems that are probably solvable such as discriminating between indoor and outdoor scenes, urban and rural scenes, etc. It should also be possible to classify people based on activity (talking, smiling, walking) and their characteristics (age, gender, hair color, facial hair, glasses, etc.). It may even be possible to robustly identify people by computer. Many research efforts are making progress in this area.

References

[1]. www.salon.com. "Bill Gates' Other CEO." 2/7/2000.

[2] www.ap.org

[3]D. P. Huttenlocher, S. Ullman. "Recognizing Solid Objects by Alignment with an Image." IJCV. 5:2, pp. 195-212, 1990.

[4] A. Pentland, B. Moghaddam, T. Starner. "View-Based and Modular Eigenspaces for Face Recognition." CVPR 1994.

[5] M. Turk, A. Pentland. "Eigenfaces for Recognition." Journal of Cognitive Neuroscience, 3:1, pp. 71-86. 1991

[6] P. N. Belhumeur, J. P. Hespanha, D. J. Kriegman. "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. PAMI. 19:7 pp. 711-720. July, 1997.

[7]. K. S. Arun, T. S. Huang, S. D. Blostein. "Least-Squares fitting of two 3-D point sets." IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI). vol. 9. pp. 698 - 700. Sept., 1987.

[8] T. Niblett. "Constructing decision trees in noisy domains." Proceedings of the second Europena working session on learning. pp 67-78. Bled, Yugoslavia.

[9]B. Schiele, J. L. Crowley. "Recognition without Correspondence using Multidimensional Receptive Field Histograms." MIT Media Lab. Tech Report 453.

[10]B. Schiele, A. Pentland. "Probabilistic Object Recognition and Localization." ICCV '99.

[11] Martial Hebert, Jean Ponce, Terrance Boult, and Ari Gross. "Report on the 1995 Workshop on 3-D Object Recognition in Computer Vision." *Object Recogntion in Computer Vision. International NSF-ARPA Workshop, Dec., '94. Leccture Notes in Computer Science, 994. pp. 1 - 18.* Springer, 1995.

[12]. Baback Moghaddam and Alex Pentland. "Probabilistic Visual Learning for Object Detection." 5th International Conference on Computer Vision (ICCV '95) (also M.I.T. Media Laboratory Perceptual Computing Section, Technology Report No. 326).

[13]. Jae S. Lim. *Two-Dimensional Signal and Image Processing*. Prentice-Hall. 1990.

[14]. Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. "Neural Network-Based Face Detection." IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 1, January, 1998, pp. 23-38.

[15] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag. 1995.

[16]Abhijit S. Pandya and Robert B. Macy. *Pattern Recognition with Neural Networks in C++*. CRC Press. Boca Raton, FL. 1996.

[17]Marvin Minsky and Seymour Papert. *Perceptrons: an Introduction to Computational Geometry* (expanded edition) MIT Press. Cambridge, MA, 1988.

[18] K-K Sung, T. Poggio."Example-Based Learning for View-Based Human Face Detection". PAMI, 20:1, pp. 39-51, 1998.

[19] Edgar Osuna, Robert Freund, and Federico Girosi. "Training Support Vector Machines: an Application to Face Detection." CVPR '97. pp. 130 - 136.

[20] A. J. Comenarez and T. S. Huang. "Face Detection with Information-Based Maximum Discrimination." CVPR '97. pp 782 - 787.

[21] P. J. Phillips, et. al. "The FERET Evaluation Methodology for Face-Recognition Algorithms." CVPR '97. pp. 137 - 143.

[22] Rutishauser, H. "The Jacobi Method for Real Symmetric Matrices." Numer. Math., 9, pp. 1-10, 1966.

[23] B. V. K. Vijaya Kumar. Lectures on Pattern Recognition. In Press.

[24]. Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks." *Machine Learning*, 20, pp 273-297. 1995.

[25]. Fredercio Girosi and Tomaso Poggio. Lecture notes for MIT Course 9.520: *Learning, Approximation, and Networks* (class 9). http: //www.ai.mit.edu/projects/cbcl/course9.520/

[26] M. C. Burl and P. Perona. "Recognition of Planar Object Classes." CVPR '96. pp 223 - 230.

[27] B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press. 1996.

[28] C. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press. Oxford, 1995.

[29] D. Casasent and L. Neiberg. "Classifier and Shift-invariant Automatic Target Recogntion Neural Networks." *Neural Networks*. Vol. 8, No. 7/8. pp. 1117 - 1129. 1995.

[30] H. Murase and S. K. Nayar. "Visual Learning and Recognition of 3-D Objects from Appearance." *International Journal of Computer Vision*. 14, pp 5-24. 1995.

[31] William M. Wells III. "Statistical Approaches to Feature-Based Object Recognition." *International Journal of Computer Vision*. 21(1/2), pp. 63-98, 1997.

[32] Y. Lamdan and H. J. Wolfson. "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme." ICCV, 1988. pp. 238 - 249.

[33] Shimon Ullman and Ronen Basri. "Recognition of Linear Combinations of Models." PAMI. 13:10, pp. 992 - . Oct. 1991.

[34] John Krumm. "Eigenfeatures for Planar Pose Measurement of Partially Occluded Objects." CVPR, 1996.

[35] K. Ohta and K. Ikeuchi. "Recognition of Multi Specularity Objects using the Eigen-window." Tech. Report. CMU-CS-96-105.

[36] M. Koch, M. M. Moya, L. D. Hostetler, R. J. Fogler. "Cueing, Feature Discovery, and One-class Learning for Synthetic Aperture Radar Automatic Target Recognition." *Neural Networks*. Vol. 8, No. 7/8. pp. 1081 - 1101. 1995.

[37] B. V. K. Vijaya Kumar. "Tutorial Survey of Composite Filter Designs for Optical Correlators." *Applied Optics*. 31:23, pp. 4773 - 4801. August, 1992.

[38] D. Roth, M-H. Yang, N. Ahuja. "A SNoW-Based Face Detector." NPPS '99.

[39]P. A. Viola. "Complex Feature Recognition: A Bayesian Approach for Learning to Recognize Objects." AI Memo No. 1591. November, 1996.

[40]B. Moghaddam and A. Pentland. "Probabilistic Visual Learning for Object Representation." PAMI. 19:7. pp. 696 - 710 July, 1997.

[41] M. D. Wheeler and K. Ikeuchi. "Sensor Modeling, Probabilistic Hypothesis Generation, and Robust Localization for Object Recognition." PAMI. 17:3. pp. 252 - 265. March, 1995.

[42] F. Stein and G. Medioni. "Structural Indexing: Efficient 3-D Object Recognition." PAMI. 14:2. pp. 125 - 144. Feb. 1992.

[43] J. Huang, S. Gutta, H. Wechsler. "Detection of Human Faces Using Decision Trees." 2nd Intl. Conf. on Automated Face and Gesture Recognition.

[44] A. N. Rajagopalan, P. Burlina, R. Chellappa. "Higher Order Statistical Learning for Vehicle Detection in Images." ICCV '99. pp. 1204 - .

[45] J. L. Mundy, A. Zisserman (eds). *Geometric Invariance in Computer Vision*. MIT Press. Cambridge, MA, 1992.

[46] D. Forsyth, et. al. "Invariant Descriptors for 3D Object Recognition and Pose." PAMI 13:10. pp. 971-991.

[47] W. E. L. Grimson (with contributions from T. Lozano-Perez and D. P. Huttenlocher. *Object*

Recognition by Computer: The Role of Geometric Constraints. MIT Press. Cambridge, MA, 1990.

[48] J. J. Koenderink, *Solid Shape*, MIT Press, 1990, Cambridge, MA

[49] H. Moravec. "Obstacle Avoidance and Navigation in the Real World by a Seeing Eye Robot Rover." Ph. D. Stanford, 1980.

[50] J. Shi and C. Tomasi. "Good Features to Track." CVPR '94. pp. 593-600. June, 1994.

[51] M. Gori, F. Scarselli. "Are Mulilayer Perceptrons Adequate for Pattern Recognition and Verification". PAMI 20:11. pp. 1121-1132. November, 1998.

[52] S. Y. Kung. *Digital Neural Networks*. Prentice-Hall, 1993.

[53] D. J. Field. "Wavelets, vision and the statistics of natural scenes". Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences. Vol. 357 Issue 1760 - 1999. pp. 2527-2542

[54] M. J. Swain, D. H Ballard, "Color Indexing." IJCV 7:1, pp. 11-32. November 1991,

[55]Suetens, Fua, Hansen. "Computational Studies for Object Recognition", ASM

[56] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 2nd Ed, 1984.

[57] D. Marr. *Vision*. Freeman Publishers, 1982.

[58]G. Strang, T. Nguyen. *Wavelets and Filter Banks*. Wellesley - Cambridge Press. Wellesley, MA. 1997.

[59]M. Vetterli, J. Kovacevic. *Wavelets and Subband Coding*. Prentice-Hall, 1995.

[60] I. Daubechies. *Ten Lectures on Wavelets*. CBMS-NSF Regional Conference Series in Applied Mathematics 61. 1992.

[61] P. C. Cosman, R. M. Gray, M. Vetterli. "Vector Quantization of Image Subbands: A Survey." IEEE Transactions on Image Processing. 5:2 pp. 202-225. February, 1996.

[62] Z. Gigus, J. Canny, R. Seidel. "Efficiently Computing and Representing Aspect Graphs of Polyhedral Objects." PAMI 13:6. pp. 542-551. June 1991.

[63] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[64] P. Domingos, M. Pazzani. "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss." Machine Learning. 29, pp 103-130. 1997.

[65] T. Kailath. *Linear Systems*. Prentice-Hall, 1980.

[66]R. Sekuler, R. Blake. *Perception* (3rd Edition). McGraw-Hill, 1994.

[67]E. B. Goldstein. *Sensation and Perception* (4th Edition). Brooks / Cole Publishing. 1996.

[68] J. Frisby. *Seeing: Illusion, Brain, and Mind*. Oxford University Press, 1979.

[69] D. H. Hubel. *Eye, Brain, and Vision*. Scientific American Library Series, #22. 1988.

[70]Y. Freund, R. E. Shapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting." Journal of Computer and System Sciences. 55:1, pp. 119-139. 1997.

[71]L. Breiman. "Arcing Classifiers." The Annals of Statistics. 26:3, pp. 801-849. 1998.

[72]R. E. Shapire, Y. Singer. "Improving Boosting Algorithms Using Confidence-rated Predictions." Machine Learning 37:3, pp. 297-336. December, 1999.

[73]E. Bauer, R. Kohavi. "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants." Machine Learning. 36:1/2, pp. 105-139. July, 1999.

[74] T. Jaakkola, M. Meila, T. Jebara. "Maximum entropy discrimination." MIT AITR-1668, 1998.

[75] C. Goerick, D. Noll, M. Werner, "Artificial Neural Networks in Real-Time Car Detection and Tracking Applications." Pattern Recognition Letters. 17:4. pp. 335-343. April, 1996.

[76] H. Schneiderman, T. Kanade. "Probabilistic Modeling of Local Appearance and Spatial Relationships for Object Recognition." CVPR '98.

[77]I. E. Gordon. *Theories of Visual Perception*. John Wiley & Sons. 1989

[78] M. Lades, J. C. Vorbruggen, J. Buhmann, J. Lange, C. v. d. Malsburg, R. P. Wurtz, W. Konen. "Distortion Invariant Object Recognition in the Dynamic Link Architecture." IEEE Transactions on Computers. 42:3. pp 300 - 311. March, 1993.

[79] L. Wiskott, J-M Fellous, N. Kruger, C. v. d. Malsburg. "Face Recognition by Elastic Bunch Graph Matching." PAMI. 19:7. pp. 775-779, 1997.

[80]K. Dowd and C. Severance. *High Performance Computing* (2nd Ed). O'Reilly, 1998.

[81] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, T. Poggio. "Pedestrian Detection Using Wavelet Templates." CVPR '97. pp. 193 - 97.

[82]C. P. Papageorgiou, M. Oren, T. Poggio. "A General Framework for Object Detection." ICCV '98.

[83] C. P. Papageorgiou, T. Poggio. “A Trainable Object Detection System: Car Detection in Static Images.” MIT AI Memo No. 180. October, 1999.

[84] T. D. Rikert, M. J. Jones, P. Viola. “A Cluster-Based Model for Object Detection.” ICCV 1999. pp. 1046 - 1053.

[85] S. J. Wan, S. K. M. Wong. “A Measure for Concept Dissimilarity and Its Applications in Machine Learning.” Proceedings of the International Conference on Computing and Information. pp. 267-273.

[86]I. Kononenko. “Semi-naive Bayesian Classifier.” Proceedings of the Sixth European Working Session on Learning. pp. 206-219.

[87]H. Rowley. “Neural Network-Based Face Detection.” Ph.D thesis. CMU-CS-99-117. 1999.