Texture-Based Statistical Models for Object Detection in Natural Images

by

Thomas D. Rikert

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

Bachelor of Science in Computer Science and Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 20, 1999
[June 1999]

Author ....
Department of Electrical Engineering and Computer Science
May 20, 1999

Certified by ....
Paul A. Viola
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Texture-Based Statistical Models for Object Detection in Natural Images

by

Thomas D. Rikert

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 1999, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Computer Science and Engineering
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Texture is an important cue for detecting objects that undergo shape deformation, pose changes, and variations in illumination. We propose a general statistical model which relies on texture for learning an object class from a set of example images. Once the model learns the distribution of the object class images, it can then be used to classify new images according to their membership in the class. The distribution of images is captured by a set of feature vectors which have been shown to produce good texture detection and synthesis results [4]. Our statistical model uses these feature vectors for classification and can handle larger variations in the appearance of the object class than previous approaches. We estimate the distribution of feature vectors for an object class by clustering the data and then forming a mixture of Gaussian model. The mixture model is further refined by determining which clusters are the most discriminative for the class and retaining only those clusters. After the model is learned, test images are classified by computing the likelihood of their feature vectors with respect to the model. We present excellent results in applying our technique to face detection and car detection.

Thesis Supervisor: Paul A. Viola
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

This research was supported by Compaq Computer Corporation's Cambridge Research Laboratory through the MIT EECS VI-A Internship Program. It was also supported by NTT Electronics Corporation through a grant to Professor Paul Viola at the MIT Artificial Intelligence Laboratory.

I would like to thank Paul Viola for his enthusiasm, good humor, and encouragement during this research and my undergraduate years at MIT. He has challenged me to think with both creativity and precision in our many conversations. Working alongside him has enriched my education with new ideas and research problems. He played a major part in making MIT a fun and exciting place for me to grow academically and personally.

I could not have asked for a better company advisor than Mike Jones at Compaq. Mike has been a great teacher and co-investigator. Many of the insights in this thesis arose from our conversations, and his examples of good technical writing have helped me in composing this thesis. In addition, the Compaq Cambridge Research Lab (CRL) has been a supportive and fun place to work each summer. Thanks to director Bob Iannucci and all of the researchers there for creating such an outstanding environment.

My academic advisor, Professor Patrick Winston, has encouraged me to pursue my goals even when the odds were against me. His confidence in my abilities means a great deal to me.

Special thanks to Jeremy De Bonet for providing several figures for this document, and L.J. Ruell and Gene Preble for collecting the face database used in this research. I am grateful to my officemate Dan Snow for provided feedback on this document. I wish him the best as he continues his MIT career. Also thanks to Linda Ungsunan and my brothers at Phi Kappa Theta for their support throughout all my years at MIT.

Finally, I want to acknowledge my Lord, Jesus Christ, for making this all worthwhile and for giving me the strength to do it. "Christ does not destroy reason; he

dethrones it." - Soren Kiekegaard

Contents

1 Introduction **11**

1.1 Problem Statement 11

1.2 Contributions 14

1.3 Document Overview 15

2 Statistical Models of Natural Images **16**

2.1 Distributions of image pixels 17

2.2 Distributions of image features 17

2.3 Density estimation 18

2.3.1 Parzen-window density estimators 18

2.3.2 Clustering 19

2.3.3 Gaussian mixture models 19

2.4 Conclusion 20

3 Texture Analysis and Synthesis **21**

3.1 Properties of image texture 21

3.2 Texture analysis 22

3.2.1 Multi-scale, multi-orientation wavelet transform 23

3.2.2 Parent vectors 25

3.3 Texture synthesis 26

3.3.1 Distance functions 27

3.3.2 Choosing thresholds 28

3.3.3 Synthesizing structured images 29

3.4 Discussion 31

4 The Statistical Model **32**

4.1 Overview 32

4.2 Estimating the distribution 34

4.2.1 Clustering algorithm 35

4.3 Discriminative analysis 37

4.4 Discussion 39

5 Classification **41**

5.1 Evaluating the mixture model 41

5.1.1 Comparing distributions: sets of vectors 43

5.2 Bernoulli trials 44

5.3 Discussion 45

6 Results **47**

6.1 Face Detection 47

6.1.1 Building the face model 48

6.1.2 Correlating clusters with perceptual features 49

6.1.3 Face classification results 50

6.1.4 Bernoulli trial results 53

6.2 Car detection 53

6.3 Image database retrieval 55

7 Conclusion **59**

7.1 Summary 59

7.2 Future Work 59

7.3 Discussion 61

References **63**

List of Figures

1-1 A jacket hanging on the wall is easily recognized by its shape. 12

1-2 A jacket tossed on the floor undergoes complex shape deformations. The patterns of different colored regions and different types of material are still recognizable, however. 12

1-3 A face with extreme variations in illumination is a difficult case for many template-based techniques. Some features, such as the right eye, are washed out and may cause a detector looking for two eyes to fail. 13

1-4 This image shows the same face after a change in pose. The structure of the face changes significantl: one eye is occluded and the lines around the nose and chin dominate the shape. The hair and skin texture is still clearly visible, however. 14

3-1 Oriented derivative filters are applied to an image to produce subbands highlighting edges and ridges at several different angles. This example has a fixed scale. See figure ?? for an example decomposition over both scale and orientation. 23

3-2 The oriented filters can also be applied to downsampled versions of the input image to produce subbands at different resolutions. 24

3-3 Illustration of parent vector structure. The grids represent an image pyramid. Each pixel is associated with a set of filter values capturing local features. The chain of line segments show the pixels whose filter values make up a single parent vector. 26

3-4 The smaller images on the left were input to the synthesis algorithm. Note that the larger synthesized images do not contain tiled versions of the original texture and instead have new patterns that are perceptually close to the original. 29

3-5 Texture synthesis examples using a set of face images as input textures. See text for an explanation. 30

3-6 Texture synthesis examples showing the effect of thresholds. See text for an explanation. 31

4-1 An outline showing construction of the model. A target class model (faces) and background model are built independently, producing a large number of clusters for each class. Discriminative analysis produces a final model with a small number of clusters drawn from both classes. 33

4-2 Snapshots of the clustering algorithm operating on randomly generated data points in two dimensions. It begins with all the data points (circles) as single population clusters. Neighboring points are then clustered together, represented by the squares. After a pass has been made through all the data, the algorithm repeats by merging neighboring clusters from the previous iteration. In this figure, the area of the boxes is proportional to the variance of the cluster, and the thickness of the box walls is proportional to the population of the cluster. Each histogram bar corresponds to the population of one cluster. 36

4-3 The discriminative analysis procedure. The images on the left axis represent the face and background training data. For the "Face Clusters" and "Background Clusters" columns, we count the number of face vectors that are near each face cluster, and then count the number that are near each background cluster. The procedure is repeated for the background image vectors. This results in four histograms, where each bar corresponds to the vector count for one cluster. We then calculate the difference between the face cluster histograms, and then the background cluster histograms. Only clusters with histogram differences greater than the threshold are retained for the model. 38

6-1 Example face images from the training set. 48

6-2 Example background images from the training set. 48

6-3 Receiver operating characteristics curve for a test set of 266 frontal faces and 2500 non-faces using the mixture of Gaussians classifier. Note that to focus on the interesting part of the graph, the correct detection rate axis begins at 50%. 50

6-4 A cluster was chosen from the face model and compared to all parent vectors in each test image. The white boxes show the postion in each image where a parent vector was near the cluster. This cluster apparently represents a lip-corner feature. 51

6-5 Results for two more face clusters. Again, the white boxes show the position in each image where a parent vector was near the cluster. The top set of images shows responses for a cheek cluster while the bottom set shows responses for an eye-corner cluster. 51

6-6 Example non-frontal face images which the face detector correctly classified. 52

6-7 Receiver operating characteristics curve for a test set of 266 non-frontal faces and 2500 non-faces. 53

6-8 Receiver operating characteristics curve for a test set of 266 frontal faces and 2500 non-faces using the Bernoulli classifier. Note that to focus on the interesting part of the graph, the correct detection rate axis begins at 50%. 54

6-9 Receiver operating characteristics curve for a test set of 118 non-frontal faces and 2500 non-faces using the Bernoulli classifier. The performance of the Bernoulli detector is much worse than the mixture model in the non-frontal case%. 54

6-10 Eight of the 24 example cars used to train the car detector. 55

6-11 Receiver operating characteristics curve for a test set of 24 side views of cars and 200 non-cars. 56

6-12 An example detection by the preliminary system 57

6-13 Notice that the center of the face has been detected, as well as a second region centered on the hair and ear. We suspect the that the training data contained enough examples of hair and ears for the model to respond to these textures in a test image. 58

6-14 Another example where more than one patch of the face was found by the model. 58

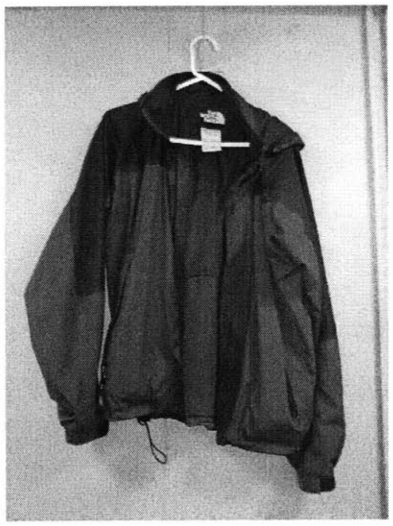

Figure 1-1: A jacket hanging on the wall is easily recognized by its shape.

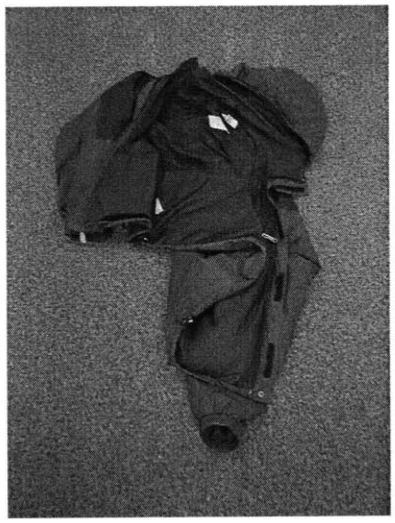

Figure 1-2: A jacket tossed on the floor undergoes complex shape deformations. The patterns of different colored regions and different types of material are still recognizable, however.

Chapter 1

Introduction

1.1 Problem Statement

This thesis is a response to an engaging scientific question that has its roots both in computational vision and visual psychology: "What underlies the human ability to recognize objects under *deformation* and changes in pose?" Take for example the recognition of a jacket casually tossed to the floor. Somehow the almost infinite variation in appearance is easily captured by the observer's model of the jacket. It seems unlikely that recognition of the jacket's image is based on purely geometric reasoning. It also seems unlikely that the observer has built a complex model of jacket deformation painstakingly acquired over many thousands of examples.

Similarly, the appearance of a human face changes drastically as it rotates from a frontal to a sidelong view. The frontal view contains several features such as the eyes, nose, and mouth. These features are arranged in a symmetric pattern which has consistent proportions in most frontal images. When the face is viewed from an oblique angle, this symmetric pattern becomes distorted or is no longer visible. The arragement of features in the image has changed and some features, such as an eye, may be occluded. Furthermore, the structure of the image may be dominated by the shape of the nose and chin. What is common to both views, however, is the *texture* of the hair, skin, and iris. A recognition system incorporating these textural features may be more successful than using structure alone.

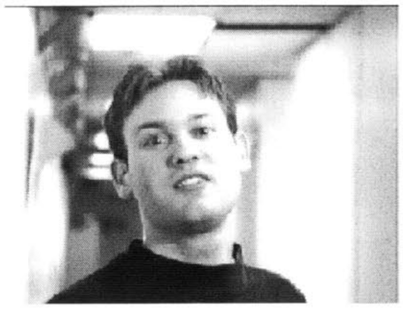

Figure 1-3: A face with extreme variations in illumination is a difficult case for many template-based techniques. Some features, such as the right eye, are washed out and may cause a detector looking for two eyes to fail.

The hypothesis of this research is that object recognition under these very difficult conditions relies on texture recognition. Motivation for the texture-based approach of this investigation has come from recent successes in texture representation and recognition [8, 4, 25, 19]. Each of these approaches can be used both to recognize texture as well as generate novel images. Based on the quality of these generated textures, it is clear that these four approaches have gone far beyond the previous state of the art in texture modeling. Taken together these new results have destroyed classical distinctions between textons, noisy textures and highly structured patterns.

We extend the ideas from texture modeling to handle typical images containing an object and background. This new general statistical model is capable of learning an object class from a set of example images and correctly classifying new images according to their membership in the class. The model represents both the textural and structural characteristics of the target object using a distribution of feature vectors which have been shown to produce good texture detection and synthesis results. The feature vectors capture the joint occurence of local features at multiple scales, and also characterize the distribution of edges and ridges at different orientations in the image.

Research findings in cognitive science provide a biological motivation for our example-based technique [13]. After viewing several examples, people can accurately group images of similar-looking objects together despite differences in pose, lighting, size, and image quality. People learn which variations are allowable, and which are

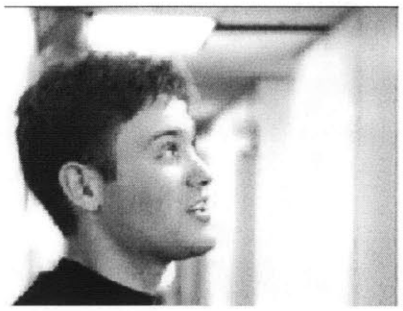
Figure 1-4: This image shows the same face after a change in pose. The structure of the face changes significantl: one eye is occluded and the lines around the nose and chin dominate the shape. The hair and skin texture is still clearly visible, however.

superfluous for recognizing the target image class . The enormous amount of variation among even familiar objects like human faces makes this a challenging and interesting problem for study in cognitive science as well as computer vision.

One can imagine several useful applications for this classification system. For example, a system robust to different pose, illumination, and image quality could search through a collection of images from the World Wide Web. Because images on the Web come from many different sources, searching techniques must handle unpredictable variations robustly. This is currently a major problem for state-of-the-art systems. Similarly, a system which runs in real-time could track objects whose shape and reflectance change as their orientation with respect to the camera changes through time.

1.2 Contributions

The main contribution of this thesis is the investigation of a new framework for object class detection based on an extended texture model (e.g. face or car detection). The advantage of this approach is that it can handle larger variations in the appearance of the object class than previous techniques. Starting from the De Bonet and Viola texture model, we will extend it to allow learning from hundreds or thousands of images. In order to do this we must eliminate much of the complexity in the density model. Complexity is removed in two ways: i) using unsupervised approaches to cluster the feature space; ii) using a supervised approach to maximize the discriminative ability of the density estimate.

The validity of the approach is tested using two different image classes: pictures of human faces and cars. We have constructed an image database of both target images and background images in order to measure how well the statistical model can find target objects in novel test images.

There are several questions this thesis addresses. First, what is a good measure of the distance between two high-dimensional feature vectors? This distance metric is critical in grouping together vectors to form a distribution. Second, can an optimal distribution for detection be extracted from the original feature vector distribution? Because noise and other image artifacts introduce errors, some peaks in the density may be distractors and decrease classification rates. Some peaks may arise from features that occur commonly in both the target class and background images, and therefore provide little information for discriminating a novel image. Thirdly, do peaks in the density correspond to perceptual features in the image, or are they related to subtle image characteristcs? Our results answer each of these questions.

1.3 Document Overview

This document is divided into seven chapters. Chapter 2 summarizes previous work on statistical models for natural images. In Chapter 3, we introduce several prerequisite concepts from texture analysis and synthesis before proceeding to discuss our statistical model in Chapter 4. Different procedures for computing the probability of a novel test image are described in Chapter 5. In Chapter 6, the experimental setup and results are presented. Finally, Chapter 7 draws conclusions from our findings and suggests interesting directions for future research.

Chapter 2

Statistical Models of Natural Images

Detecting objects in images is difficult because future image inputs are not known perfectly at the time of system design, and the space of all possible images is enourmous. The future inputs can only be characterized in terms of their "typical" or "likely" behavior using some *statistical* model. A statistical modeling approach can provide a principled mechanism for learning which properties of images are important for recognition and which are not by comparing statistical measurements from different images. Statistical object recognition is a means of handling, or perhaps ignoring, the wide variations that are observed in natural images. This chapter reviews previous work in statistical models and lays the groundwork for our texture-based approach.

We use the term "natural images" to describe pictures taken of the physical world using the visible light spectrum, and at a scale accessible with an everyday camera. We do not, for example, consider synthetic images such as computer graphics or photographs taken in the infrared spectrum. We suspect that these images will have different statistical distributions of pixels and features than images commonly found in databases or on the World Wide Web.

2.1 Distributions of image pixels

Many of the strongest results of statistical modelling are in face detection and recognition. One criticism of these approaches, however, is that they attempt to model the entire image as a single rigid patch – making it difficult to model changes in pose and feature location [22, 21, 15]. More recently, these technique have been generalized to include schemes for modelling deformations in the image plane [1, 9, 6, 11]. These techniques not only learn a set of allowed variations in the image values, but also a set of allowed variations in pixel location. Nevertheless reliable detection and recognition of images across a wide variety of images is not yet a solved problem.

A number of more general statistical models of recognition have also been proposed: [23, 17, 2]. These attempt to model the appearance of localized features and then separately model feature location. While these are steps in the right direction, these approaches often require a great deal of compute time both for model learning and for image recognition.

2.2 Distributions of image features

Most previous approaches attempt to model the distribution of images directly. Our approach instead attempts to model the distribution of multi-scale features. von der Malsburg and colleagues have shown excellent results on face recognition using a similar set of multi-scale features [10, 24]. Recently, Papageorgiou *et al.* have proposed multi-scale features for object class detection [12]. They use Haar wavelets computed at particular positions in the image to form feature vectors for a support vector machine classifier. Our approach is much more radical in its disregard for feature location.

Schiele and Crowley's work on multidimensional receptive field histograms [16] also has some similarities to our work. They also look at the distribution of feature vectors formed from filter responses at different scales. Their focus differs in that they are looking at building models of single objects as opposed to object classes.

Also their feature vectors do not use the multi-scale, multi-orientation feature vector structure that we use. Furthermore, they use fairly low dimensional feature vectors which makes the use of histograms practical.

2.3 Density estimation

We have chosen to model the distribution of multi-scale image features. How can we construct the distribution? As we noted previously, image pixel distributions and image feature distributions are difficult to model directly and are not always well-approximated by parametric distributions. According to Duda and Hart [5], most practical problems involve multimodel densities which are nonparametric. There are several common techniques for estimating nonparametric densities from example data.

2.3.1 Parzen-window density estimators

De Bonet and Viola [2] have been successful at modeling image textures by building nonparametric distributions of vectors using a simple Parzen-window density estimator. Preliminary experiments for this thesis used a Parzen-window density estimator for grouping together similar multi-scale feature vectors. The basic idea is to place a hypercube cell around each sample in space, and see how many other data points fall within the cell. If we let h_n be the length of an edge of a hypercube in our density space, then its volume is given by

$$V_n = h_n^d \tag{2.1}$$

If we define a window function as

$$\phi(\mathbf{u}) = \begin{cases} 1 & |u_j| \leq \frac{1}{2} \qquad j = 1, \ldots, d \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

then $\phi(\mathbf{u})$ defines a unit hypercube centered at the origin. Thus, the number of samples in this hypercube is given by

$$k_n = \sum_{i=1}^{n} \phi(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}) \tag{2.3}$$

and averaging n samples over the volumn V_n we obtain the estimate

$$p_n(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \phi(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}) \tag{2.4}$$

2.3.2 Clustering

We found clustering to produce good distributions in a flexible and reasonably efficient manner. The particular algorithm used for this research is presented in Chapter 4. In simplest terms, clustering is grouping similar data points together. A clustering procedure *summarizes* the data by describing groups of samples points instead of the individual points themselves. We would expect that samples in the same cluster will be more similar than samples in different clusters.

A distribution can be built from clusters by tracking the individual statistics of each cluster. For example, we can keep track of the mean and variance of a cluster's population or coordinates. A critical component of clustering algorithms is the distance function which measures similarity between data points. In high dimensional spaces, it is difficult to visualize distances between data points, so an error criterion such as the sum-of-squared error can be used to form minimum variance partitions.

2.3.3 Gaussian mixture models

Mixture models have often been applied to unsupervised learning problems where we are given unlabeled sample points [5]. They arise when an observation $\mathbf{x}$ is believed to obey the probability law $P(\mathbf{x}) = \sum_i p(\mathbf{x}|\omega_i)P(class = i)$. It may be convenient and computationally efficient to approximate $p_c(\mathbf{x}|\omega_i)$ with a parametric form. The Gaussian parametric density is often used because of its tractability.

Fitting a mixture of Gaussian models is closely related to clustering. A multidimensional Gaussian kernel may be placed at the mean of each cluster in high-dimensional space, with variance in each dimension equal to the variance of the cluster in that dimension. The set of Gaussian kernels forms a mixture model probability distribution. One example of this technique appears in [14].

2.4 Conclusion

Statistical models have emerged as a powerful approach to handling large variations in images. Some techniques have modeled the distribution of pixel values, while others have modeled vectors encoding features in the image. Density estimation from observed data lies at the heart of statistical model building, and we outlined three techniques that have been applied to image analysis: Parzen windows, clustering, and mixture models. These concepts will be used in the coming chapters. Chapter 3 will discuss the application of Parzen windows to texture synthesis, while clustering and mixture models will be presented as part of our statistical model in Chapter 4.

Chapter 3

Texture Analysis and Synthesis

As noted in Chapter 1, we suspect that texture may be an important cue for detecting objects through shape deformation, pose changes, and variations in illumination. This chapter reviews the properties of image texture that may assist in detection, and techniques for analysis and synthesis of texture. This background is important since the feature vectors used to construct our statistical model in the next chapter are the same feature vectors which have been shown to produce excellent synthesis results.

Our framework follows De Bonet and Viola [4] in that it models the distribution of "parent vectors" obtained from many training images. The parent vectors are a collection of filter responses at different scales of a steerable image pyramid [18]. Understanding how parent vectors are constructed and how they have been applied to texture analysis and synthesis will provide intuition on why they are useful for our detection task.

3.1 Properties of image texture

A texture can be viewed as a sample from a probabilistic distribution. In other words, the texture is generated from a stochastic process that outputs some pattern subject to random perturbations. Two example textures generated by the same process will appear similar to a human observer, even when the examples have obvious differences in shape and color. We desire a similar "perceptual invariance" property for our object detection system. When an object is viewed from different angles or under different illumination, the pattern of shape and color also changes.

The idea of *stationarity*, or spatial invariance, characterizes this property that we wish to model. The distribution of features may be stationary such that features depend only on relative spatial position. Texture patches generated by the same stochastic process appear similar to a human observer even though the exact arrangement of the features may be different in each patch. By considering a structured image as a texture, we can still recognize the image even when its constituent features have been spatially rearranged or some features are missing. Recall the example in Chapter 1, where a face rotates from a frontal to profile view. The spatial location of features, such as the eyes, changes and at some point one eye becomes occluded. Nevertheless, the general patterns in the face (appearance of eyes, nose, mouth, chin) are still evident and can be used for detection.

Another property that can be used to characterize texture is the *joint occurence of local features at multiple resolutions.* Texture is particularly interesting because features appearing at low resolutions influence features appearing at higher resolutions. Analyzing the joint occurence of features across different resolutions reveals the structure of a texture in the form of cross-scale dependencies. Two textures that contain the same object should have similar cross-scale dependencies. This property will also help us match objects in the model to objects in a test image.

3.2 Texture analysis

There has been significant work on filtering a texture to characterize the properties discussed above. Several multi-scale transforms have proven useful for decomposing an input texture into subbands which reveal cross-scale correlations [18]. We have chosen the steerable wavelet transform, also known as the "steerable wavelet pyramid."

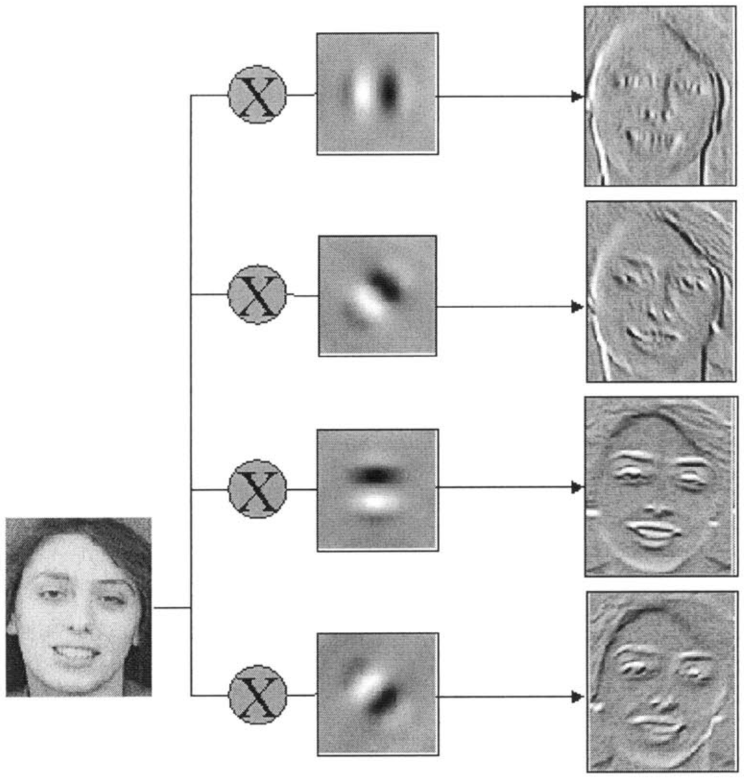

Figure 3-1: Oriented derivative filters are applied to an image to produce subbands highlighting edges and ridges at several different angles. This example has a fixed scale. See figure 3-2 for an example decomposition over both scale and orientation.

3.2.1 Multi-scale, multi-orientation wavelet transform

Multi-scale wavelet models have proven to be an effective technique for modeling natural images. They assume that the underlying stochastic processes generating the image are statistically independent. This is a reasonable assumption since the coefficients of wavelet transformed images are uncorrelated and low in entropy. For texture, however, we realized that significant cross-scale dependencies exist. The coefficients on the same scale may be independent, but we would expect that coefficients across scales are not independent. To capture this dependency, we will use the notion of *parent vectors*, discussed in the next section.

It is also interesting to note that we defined texture as the result of a single stochastic process. We can then imagine a natural image as the sum of many independent stochastic processes, each contributing different textures to the image.

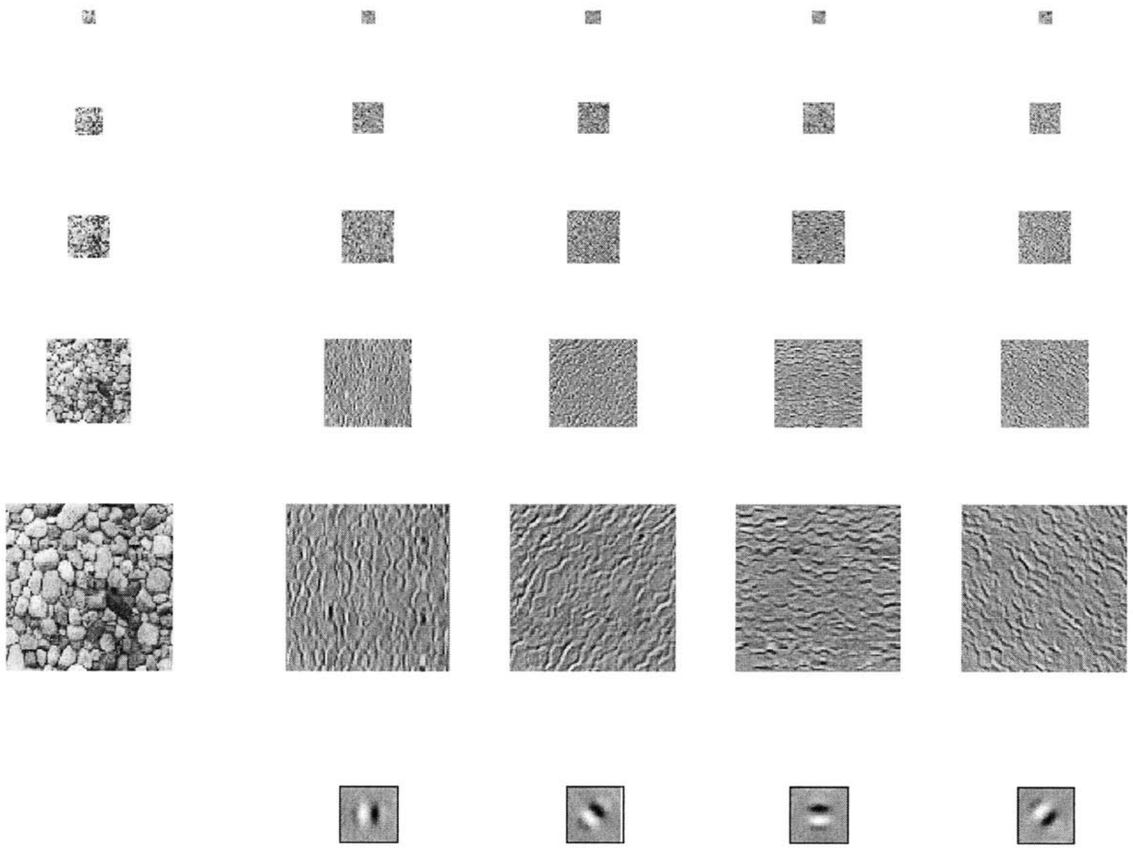

Figure 3-2: The oriented filters can also be applied to downsampled versions of the input image to produce subbands at different resolutions.

Figure 3-1 shows an image of a face transformed by a bank of oriented filters into a series of subbands. Figure 3-2 shows the combination of multi-orientation and multi-scale filtering where the oriented filters have been applied to a recursively downsampled version of the original image. We have found this transform to produce good feature vectors. The hypothesis is that images which are perceptually alike have similar distributions of features over the subbands of orientations and frequencies. Schemes for recognizing textures [2, 4, 5] and denoising images [8] have demonstrated success using this "steerable pyramid."

3.2.2 Parent vectors

Parent vectors are computed from the steerable pyramid wavelet transform. We will use the same notation as in [4]. First, a Gaussian pyramid is created from an input image I: $G_0 = I$, $G_1 = 2\downarrow(g \otimes G_0)$ and $G_{i+1} = 2\downarrow(g \otimes G_i)$, where $2\downarrow$ downsamples an image by a factor of 2 in each dimension and g is a low pass filter. At each level of the pyramid, a series of filter functions are applied: $F_j^i = f_i \otimes G_j$, where the f_i's are oriented derivative filters. For every pixel in an image define the *parent vector* of that pixel:

$$\begin{aligned}\vec{V}(x,y) = \Big[&F_0^0(x,y), F_0^1(x,y), \ldots, F_0^N(x,y),\\ &F_1^0(\lfloor\tfrac{x}{2}\rfloor,\lfloor\tfrac{y}{2}\rfloor), F_1^1(\lfloor\tfrac{x}{2}\rfloor,\lfloor\tfrac{y}{2}\rfloor), \ldots, F_1^N(\lfloor\tfrac{x}{2}\rfloor,\lfloor\tfrac{y}{2}\rfloor), \ldots\\ &F_M^0(\lfloor\tfrac{x}{2^M}\rfloor,\lfloor\tfrac{y}{2^M}\rfloor), F_M^1(\lfloor\tfrac{x}{2^M}\rfloor,\lfloor\tfrac{y}{2^M}\rfloor), \ldots,\\ &F_M^N(\lfloor\tfrac{x}{2^M}\rfloor,\lfloor\tfrac{y}{2^M}\rfloor)\Big]\end{aligned} \tag{3.1}$$

where M is the top level of the pyramid and N is the number of features.

As depicted in figure 3-3, each parent vector corresponds to exactly one pixel in the original image, and stores the wavelet coefficient for each scale and orientation at that pixel location. The high-pass and low-pass residual values are also included in the vector.

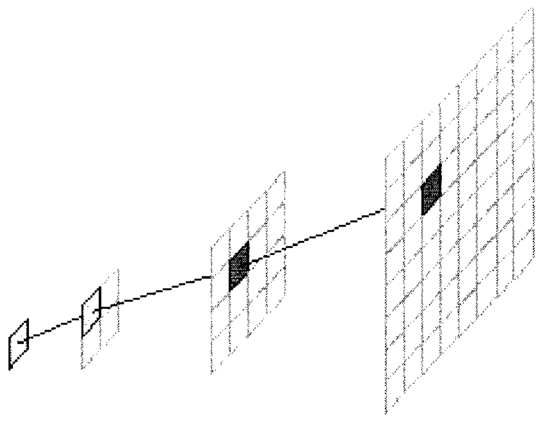

Figure 3-3: Illustration of parent vector structure. The grids represent an image pyramid. Each pixel is associated with a set of filter values capturing local features. The chain of line segments show the pixels whose filter values make up a single parent vector.

De Bonet and Viola's texture recognition work modeled the distribution of parent vectors using a Parzen window density esimator. The Parzen density estimator, while quite flexible, requires time proportional to the quantity of training data. In our experiments we will use over 1000 training images. If trained in a naive fashion, the resulting density estimator would require many minutes to evaluate. Instead of the Parzen window model, we will use a clustering algorithm to find significant clusters of parent vectors from hundreds of training images. From these clusters a mixture of Gaussian model is used to approximate this distribution of parent vectors. Approximating the distribution is the focus of Chapter 4.

3.3 Texture synthesis

We have established that texture plays an important role in detection. Synthesis is a means for visualizing the feature vector distribution we are attempting to construct. It is worthwhile to understand texture synthesis since it reveals how well our feature vector representation is capturing the signature properties of the texture. Using our analysis results, we can build a model of the underlying stochastic process generating a particular texture. This model can then be used to generate synthetic texture perceptually similar to the original. This section brielfy describes techniques for texture synthesis.

The synthesis procedure also reveals which areas of the images are considered similar by the distance function. If the synthesis results are reasonable to a human observer, then we have confidence that the distance function is grouping together parent vectors which contribute to similar perceptual features.

Texture has an interesting property: it contain regions which differ by less than some discrimination threshold, and randomization of these regions does not change the perceived characteristics of the texture. In other words, at some low resolution texture images contain regions whose difference measured by some distance function is small. Reorganizing these low frequency regions, while retaining their high frequency detail will not change its textural characteristics yet will increase its visual difference [3]. De Bonet takes advantage of this property for synthesizing new textures. Rearranging feature location at low resolutions while retaining their high resolution structure corresponds to moving whole textural units.

There have been several recent successes in texture representation and recognition in addition to De Bonet. [8, 4, 25, 19]. The Heeger and Bergen technique is perhaps the most efficient of the four, but it has some difficulty in modelling highly structured patterns. While Zhu, Wu and Mumford is the most formal and well grounded of the four, it currently lacks an efficient algorithm for learning and recognition. The De Bonet and Viola approach combines the efficiency and simplicity of the Bergen and Heeger model with the modeling power of Zhu, Wu and Mumford.

3.3.1 Distance functions

In Chapter 4, we will build our parent vector distribution using clustering. We can influence how clusters are constructed by choosing a distance function which reflects our intuition of what vectors should be similar and which should be dissimilar. Unfortunately it is difficult to visualize vector proximity in the high dimensional feature space. One way to overcome this problem is to test distance functions in the synthesis task, and find which distance function results in the most reasonable synthesis results according to a human observer. If the synthesized images looked reasonable, then the distance function is probably matching vectors coming from corresponding features.

We experimented with several distance functions for texture synthesis within the DeBonet framework. The function **near()** checks if all the dimensions of the vectors satisfy the distance function **D()**. The following **D()** gave the best results for synthesis, and also produced clusters which gave the best results in detection experiments. The function measures the distance between two vectors by normalizing the absolute distance by the sum of the absolute values of each component.

```
near($\mathbf{v}_1, \mathbf{v}_2$) {
    flag = 1
    for k=1 to d
        if $D(\mathbf{v}_1[k], \mathbf{v}_2[k]) > T_k$ then
            flag = 0
    return flag
}
```

where d is the dimension of the parent vectors, $\{T_k\}$ is a set of thresholds and

$$D(\mathbf{v}_1[k], \mathbf{v}_2[k]) = \frac{|\mathbf{v}_1[k] - \mathbf{v}_2[k]|}{|\mathbf{v}_1[k]| + |\mathbf{v}_1[k]| + 1}. \tag{3.2}$$

3.3.2 Choosing thresholds

The distance function can be parameterized by a set of adjustable thresholds. One possibility for setting detection parameters is to reuse the same values that produce good synthesis results. This follows our reasoning for choosing a distance function. There is the added complexity that we must choose a combination of thresholds. We may use different thresholds for different frequencies and different orientations, or use the same threshold for all vector components.

Choosing thresholds is difficult without any prior information. If the threshold is too large, then vectors will be grouped together coursely and fine variations in the

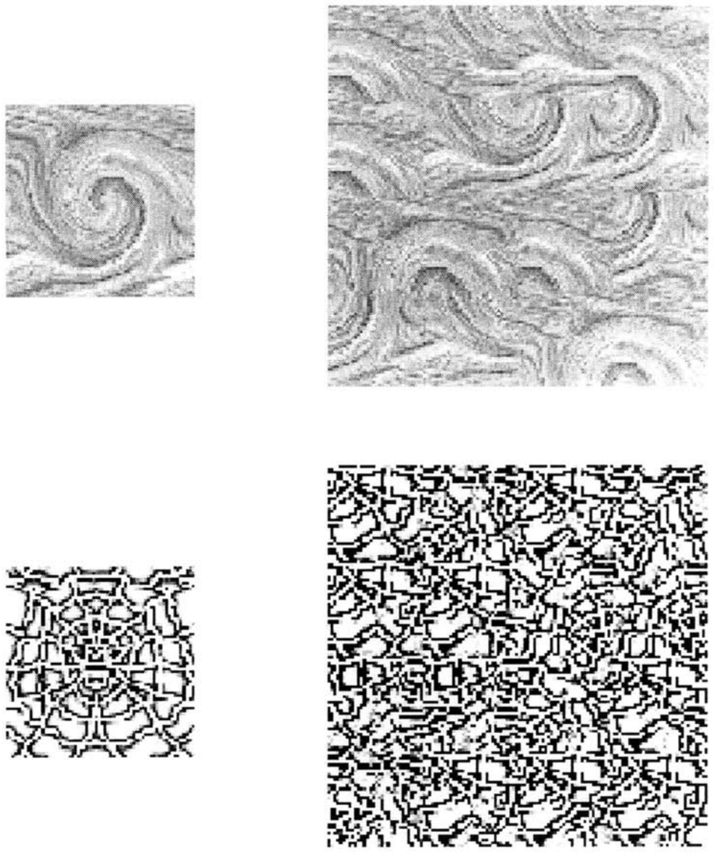
Figure 3-4: The smaller images on the left were input to the synthesis algorithm. Note that the larger synthesized images do not contain tiled versions of the original texture and instead have new patterns that are perceptually close to the original.

distribution are not resolved. If the threshold is too small, then vectors which are actually quite similar remain separated and the clustering algorithm is unable to find patterns which summarize the data. Either situation results in a distribution that does not accurately reflect the true feature vector distribution. Figure 3-4 shows two examples of texture synthesis using De Bonet's technique with reasonable thresholds and the distance function in equation 3.2.

3.3.3 Synthesizing structured images

In this investigation, we are interested in understanding texture as a means to recognize structured images of a particular class. To find which regions of a structured image have similar texture, we input face images to this texture synthesis framework

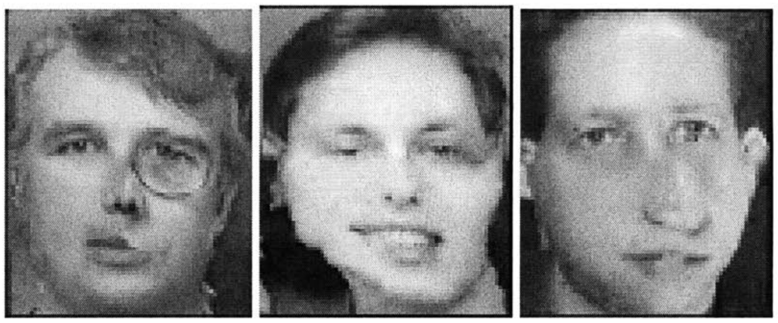

Figure 3-5: Texture synthesis examples using a set of face images as input textures. See text for an explanation.

and examined the outputs. Figure 3-4 shows some examples. Each of these images was synthesized from a texture model built from many face images as described in [3]. Notice how features (such as eyes or nose) in one face are replaced by corresponding features from another face. From this figure it appears that the parent vector representation does capture some important structure even for face images.

To help illustrate the influence of thresholds on grouping parent vectors, figure 3-6 shows three synthesis results. In (a), the thresholds for low-frequency components of the parent vectors were set very larger, while all other thresholds are small. Thus, the synthesis algorithm is free to rearrange the low frequency components but leaves the high frequency components almost identical to the original image. Note that a right-half chunk of the face overlaps a left-half chunk, aligned by the eye region. This indicates that the distance function found parent vectors from the right eye similar to the left eye. We conclude that this distance function is satisfying because it is matching vectors from corresponding perceptual features. Similarly in (b), only the middle frequency thresholds are large, so smaller textural blocks are rearranged. Notice again that within each block, all perceptual features are coherent. Finally, (c) shows an image where only high frequency thresholds are large. The high frequency detail of the image is lost while the overall structure is preserved.

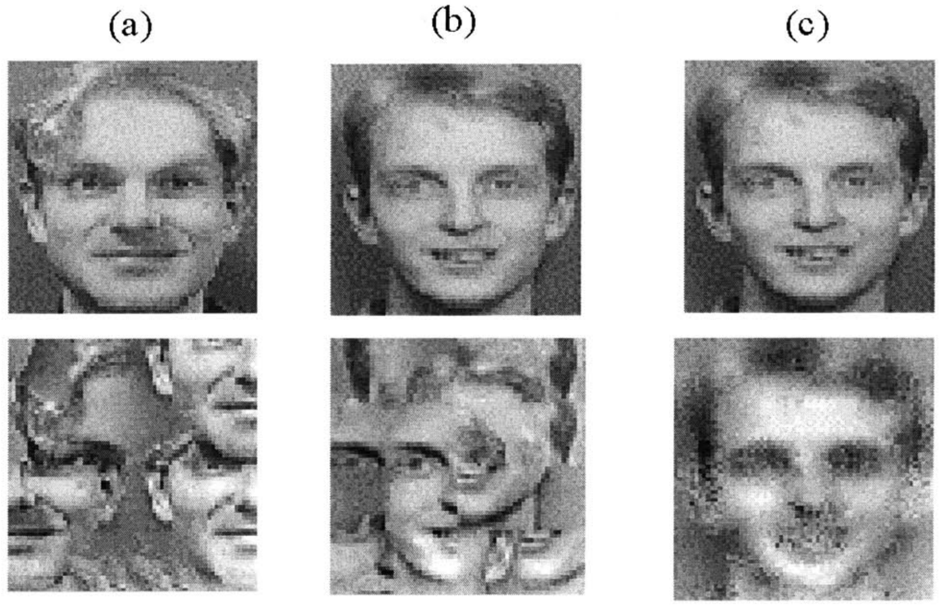

Figure 3-6: Texture synthesis examples showing the effect of thresholds. See text for an explanation.

3.4 Discussion

Texture has interesting properties that provides important cues for detecting objects in images despite shape deformation or occlusion. Parent vectors capture important properties of texture, such as the joint occurence of local features at multiple resolutions. They can also measure the distribution of edges and ridges in an image across different orientations. The stationarity property of texture allows us to rearrange parent vectors to introduce visual variation while maintaining the overall appearance of the original texture. Finally, texture synthesis is a testing ground for finding distance functions and thresholds that produce good distributions for detection.

Chapter 4

The Statistical Model

Drawing on the idea of parent vectors presented in Chapter 3, this chapter describes in detail our new statistical model for detecting objects in images. The focus is on constructing the model from many example images. Evaluating a novel test image using the model is treated as a separate topic in Chapter 5.

4.1 Overview

The basis of our framework is using a mixture of Gaussian model to estimate the distribution of parent vectors from a large set of example images of an object class.

Our method for building a model can be divided into four steps:

1. Apply multi-scale, multi-orientation filters to the training images to produce parent vectors.

2. Find clusters of similar parent vectors.

3. Find an optimal subset of clusters for detection.

4. Build the final model as a mixture of multidimensional Gaussian kernels centered on each cluster, with weight proportional to the population in the cluster and mean and variance proportional to the mean and variance of the cluster.

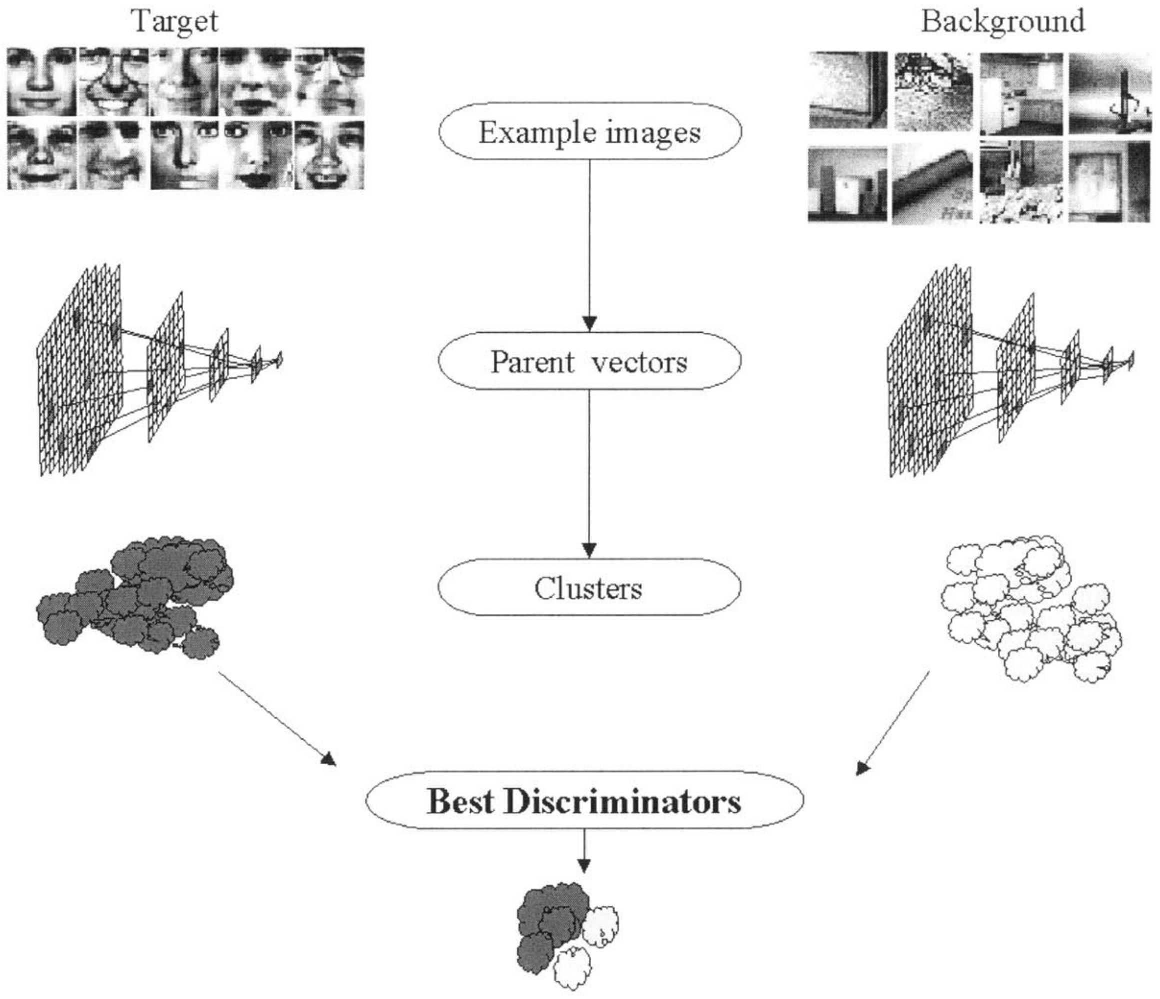

Figure 4-1: An outline showing construction of the model. A target class model (faces) and background model are built independently, producing a large number of clusters for each class. Discriminative analysis produces a final model with a small number of clusters drawn from both classes.

The same procedure is used to build an out-of-class (or background) model. The out-of-class model is combined with the in-class model using Bayes' rule to yield the probability of in-class given a parent vector. To classify a test image, the parent vectors of the test image are computed. Then the average probability of the parent vectors from the test image is computed. If this percentage is above some threshold, the test image is classified as in-class. We have also experimented with another method for evaluating the probability of a test image using histogram statistics. It is discussed in detail in Chapter 5 along with the mixture model evaluation procedure.

Beginning with the clustering algorithm, each of the steps in building a model is described in detail below.

4.2 Estimating the distribution

The model we build to represent an object class is an estimate of the distribution of parent vectors from a set of example images of the object class. To estimate the distribution of parent vectors, a clustering algorithm is first run on the data and then the resulting clusters can be used to build a mixture of Gaussian model as in [14].

We have found the standard k-means algorithms [7] for clustering to be too computationally expensive for our purposes. The problem is we have many data points and the data requires many clusters to estimate it well. For example, we have a training set of over 1 million parent vectors in a 26 dimensional space that requires hundreds of thousands of clusters to represent it well. To speed things up, we can take advantage of the fact that we have a good idea of the maximum distance that should be allowed between data points belonging to the same cluster. Based on experience with a wide variety of images, we have determined a set of distance function thresholds that produce good synthesis results. Using this distance, we take a bottom-up approach to clustering which starts with every data point as its own cluster and then combine clusters which are close together according to our distance function. We use the same **near()** function that worked well for synthesis, show below for convenience, to determine if two parent vectors are close.

```
near($\mathbf{v}_1, \mathbf{v}_2$) {
    flag = 1
    for k=1 to d
        if $D(\mathbf{v}_1[k], \mathbf{v}_2[k]) > T_k$ then
            flag = 0
    return flag
}
```

where d is the dimension of the parent vectors, $\{T_k\}$ is a set of thresholds and

$$D(\mathbf{v}_1[k], \mathbf{v}_2[k]) = \frac{|\mathbf{v}_1[k] - \mathbf{v}_2[k]|}{|\mathbf{v}_1[k]| + |\mathbf{v}_1[k]| + 1}. \tag{4.1}$$

4.2.1 Clustering algorithm

The following pseudo code describes our clustering algorithm:

```
N = number of parent vectors
M = number of clusters
Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ be the list of N d-dimensional parent vectors

M = N;
Make a cluster for each parent vector with mean $\mu_i = \mathbf{v}_i$ and variance $\sigma_i^2 =$
0
do {
    for i=1 to M-1
        for j=i+1 to M
            if near($\mu_i, \mu_j$) then {
                combine cluster i and j
                (keeping track of the
                mean, variance and
                population count)

                M = M - 1
            }
} until the decrease in M is insignificant
```

The resulting clusters are used to build a Gaussian mixture model to generalize the distribution by placing a multidimensional Gaussian kernel at the center of each cluster. Popat and Picard [14] use a similar procedure to model distributions obtained from images although their features are different from our parent vectors. The probability of vector $\mathbf{v}$ in the model for class C is given by

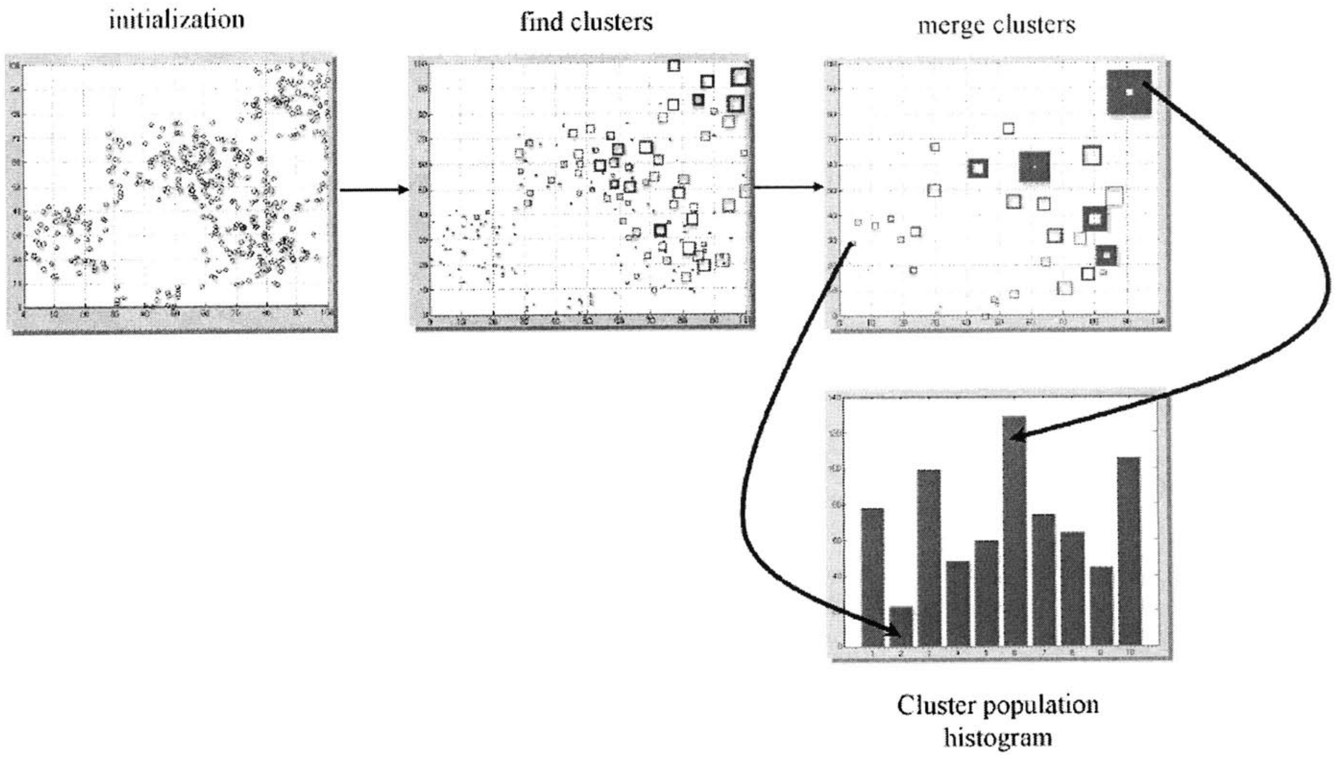

Figure 4-2: Snapshots of the clustering algorithm operating on randomly generated data points in two dimensions. It begins with all the data points (circles) as single population clusters. Neighboring points are then clustered together, represented by the squares. After a pass has been made through all the data, the algorithm repeats by merging neighboring clusters from the previous iteration. In this figure, the area of the boxes is proportional to the variance of the cluster, and the thickness of the box walls is proportional to the population of the cluster. Each histogram bar corresponds to the population of one cluster.

$$P_C(\mathbf{v}) = \sum_{m=1}^{M} w_m \prod_{i=1}^{d} k_{m,i}(\mathbf{v}[i]) \tag{4.2}$$

where the kernel k is the one-dimensional Gaussian

$$k_{m,i}(\mathbf{v}[i]) = \frac{1}{\sqrt{2\pi\hat{\sigma}_{m,i}^2}} e^{-\frac{(\mathbf{v}[i]-\mu_{m,i})^2}{2\hat{\sigma}_{m,i}^2}} \tag{4.3}$$

and $w_m = Pop_m/N$ where Pop_m is the number of parent vectors in cluster m.

4.3 Discriminative analysis

The clustering algorithm eventually produces a set of feature clusters. If the number of parent vectors in the training set is large (say over one million) then the number of clusters will often also be large (sometimes hundreds of thousands). We wish to reduce the number of clusters for two reasons: 1) to improve the accuracy of the model by keeping only clusters which discriminate between in-class and out-of-class parent vectors and 2) to increase the speed of evaluating test images by having fewer clusters to examine.

We use discriminative analysis to achieve this. The idea is to take a set of in-class and out-of-class training images and create two histograms showing how many times parent vectors from each set fall near an in-class cluster. Then we keep only those clusters which have a significantly larger count for in-class parent vectors than for out-of-class parent vectors. This reduces our in-class model to the most discriminative clusters. We can use the same method to reduce the number of clusters in both the face model and the non-face model.

There are many possible tests to determine whether the count for in-class parent vectors is "significantly" more than the count for out-of-class vectors. The test we are currently using is as follows. Let $hist_C[i]$ be the count of in-class parent vectors which are near cluster i. Let $hist_{\bar{C}}[i]$ be the count of out-of-class parent vectors which

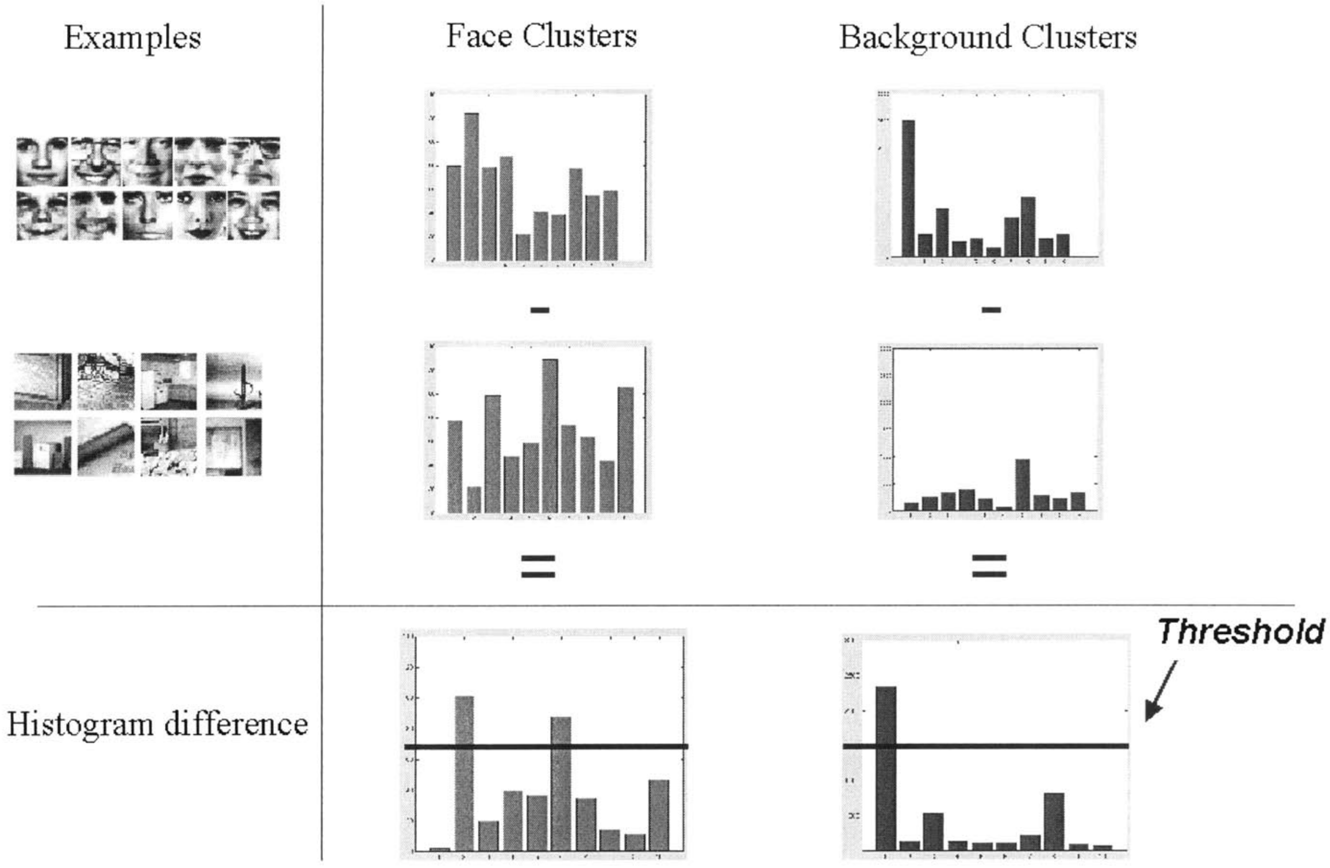

Figure 4-3: The discriminative analysis procedure. The images on the left axis represent the face and background training data. For the "Face Clusters" and "Background Clusters" columns, we count the number of face vectors that are near each face cluster, and then count the number that are near each background cluster. The procedure is repeated for the background image vectors. This results in four histograms, where each bar corresponds to the vector count for one cluster. We then calculate the difference between the face cluster histograms, and then the background cluster histograms. Only clusters with histogram differences greater than the threshold are retained for the model.

are near cluster i. Then if

$$\frac{hist_C[i]}{hist_C[i] + hist_{\bar{C}}[i]} > \Theta \tag{4.4}$$

then cluster i is a discriminative cluster and therefore retained. The threshold Θ can be any real number between 0 and 1. We used a value of 0.8 in the experiments presented in Chapter 6

Figure 4-3 illustrates the discriminative analysis idea. The images on the left axis represent training data. The training image parent vectors are compared to both the target class clusters and background class clusters. Each histogram bar represents a cluster, with height proportional to the number of training vectors that are close to the cluster center in feature space. Clusters that have similar response to both sets of training data do not help discriminate the class membership of the image, so those clusters are thrown out of the model. A threshold can be set to decide whether the difference in response for the two classes is great enough to help discriminate. In figure 4-3, the clusters corresponding to the histogram bars above the threshold will be kept in the discriminative model, while the clusters whose histogram bars are below the threshold are discarded.

4.4 Discussion

Our statistical model clusters parent vectors from many example images using a modified k-means algorithm. This is in constrast to De Bonet and Viola, who build their distribution of parent vectors using a Parzen density estimator. The Parzen density estimator requires time proportional to the quantity of training data to evaluate a new test image. Our clustering algorithm, on the other hand, summarizes the distribution using a small number of clusters in order to increase the efficiency of evaluation. We chose to use a mixture of Gaussian model on top of the clusters in order to generate a smooth and tractable density estimate for evaluating test images.

The discriminative analysis step in our model-building process is one of the most interesting products of this research. It improves the accuracy of the model by keeping only clusters which discriminate in-class and out-of-class parent vectors. It also increases the speed of evaluating test images by having fewer clusters to examine. Discriminative analysis can be viewed as a kind of filter which improves the signal to noise ratio in the distribution we have built from many noisy example images.

Chapter 5

Classification

Given our statistal model in Chapter 4, how do we decide whether a novel image contains the target object? What is the trade-off between high detection rates and the number of misclassifications? This chapter presents two different classification procedures which compute the probability that a test image contains the target object. Chapter 6 presents the experimental results of these procedures.

We found that the mixture of Gaussian model gives the best detection rates but is expensive to compute. The second approach gives nearly as good results using only histogram statistics, and is significantly less costly to compute.

5.1 Evaluating the mixture model

The previous chapters have described how we build distributions of parent vectors from example in-class and out-of-class images. We would expect to get the best classification performance by using information from *both* distributions in our evaluation procedure. The probability of a parent vector in a single class mixture of Gaussian model is computed using equation 5.1. To exploit information from both in-class and out-of-class mixture models, we combine them using Bayes rule. The in-class and out-of-class mixture models for a class C give $P(\mathbf{v}|C) = P_C(\mathbf{v})$ and $P(\mathbf{v}|\bar{C}) = P_{\bar{C}}(\mathbf{v})$.

We can use Bayes rule to yield

$$P(C|\mathbf{v}) = \frac{P(\mathbf{v}|C)P(C)}{P(\mathbf{v}|C)P(C) + P(\mathbf{v}|\bar{C})P(\bar{C})}. \tag{5.1}$$

This equation gives the probability of class C given a single parent vector $\mathbf{v}$.

The priors $P(C)$ and $P(\bar{C})$ can be chosen arbitrarily such that $P(C) + P(\bar{C}) = 1$. The choice does not effect the receiver operating characteristics curve which expresses the relationship between correct detections and false positives for classifying parent vectors [7].

Although the Gaussian mixture model has good performance (as demonstrated in Chapter 6), it also has several unsatisfying properties:

1. The final mixture model is in fact *not* an accurate approximation of the true distribution of parent vectors. Recall that part of the model building process is to throw out the clusters which are not good discriminators. The final mixture model is constructed using only a small subset of the all the clusters representing the distribution. Consequently, the distribution using all the clusters and the distribution using the subset of clusters may have very different shapes. In essence, we do not refine the mixture model in the sense of improving its ability to model the actual distribution of parent vectors. We instead reshape the distribution so it has the best discriminative power.

2. Because the best discriminating distribution may not accurately represent the true parent vector distribution, the mixture model is not a useful distribution for synthesizing new examples of the target class. We confirmed this hypothesis with synthesis experiments under the De Bonet framework. If we used the mixture model distribution as the example texture distribution, the synthesis outputs were noisy and did not contain any recognizable features.

3. The Gaussian function is expensive to compute, and the computation must be performed for each test vector. For example, a 32×32 pixel test image requires 1024 Gaussian evaluations. This may take several seconds on a Pentium II class

machine, which makes the evaluation method impractical for searching through an image database.

An alternative technique which does not use a mixture model is described in 5.2. Before this discussion, however, we need to understand how sets of parent vectors are evaluated.

5.1.1 Comparing distributions: sets of vectors

The question remains of how to use this probability model to determine if a *set* of parent vectors from a test image are more likely to come from an in-class image than an out-of-class image. De Bonet and Viola [4] suggest comparing the distribution of parent vectors from a test image against the in-class model distribution using the Kullback-Liebler (KL) divergence. The problem with this idea in our case is the distribution from a single test image will probably not look like the distribution from a large set of training images. The reason is the parent vectors from an image are not independent of each other. They form a tree structure. The model distribution is more like a collection of these trees than like a single tree. Thus, the two distributions will probably not be similar.

Instead of comparing distributions, we have used the simple idea of calculating the average probability of C given each of the parent vectors in a single test image and then thresholding this value to classify the image. Thus, if

$$\frac{\sum_{j=1}^{N} P(C|\mathbf{v}_j)}{N} > t \tag{5.2}$$

for some threshold t then the image is classified as belonging to C, otherwise it is classified as $\bar{C}$.

The average probability can be viewed as one of many possible statistics we could calculate. If we estimate the distribution of a statistic for the training data, we could then evaluate the likelihood of this statistic computed for a test image. Using multiple statistics to evaluate test images could produce more accurate tests for a classifier. The next section discusses alternative statistics for evaluating the probability of a test image.

5.2 Bernoulli trials

The mixture model classification requires the evaluation of many Gaussian density functions. An alternative approach is to examine a statistic of the clusters directly. Ideally, we would like a scalar-valued statistic which is fast to compute and has good discriminating power.

One approach is to imagine each cluster as casting a "vote" about the class membership of a particular parent vector. The vote can be either for in-class membership or out-of-class membership. We constrain the cluster votes to be independent in order to simplify the probability calculation. Under these two constraints, the "vote" is like a flip of a biased coin, and can be modelled by a Bernoulli trial. The bias of the in-class coin is the in-class response of the cluster normalized by the sum of the in-class and out-of-class response, similar to the structure of the Bayes' rule formulation in equation 5.1.

The bias of the coin is determined by two statistics: the number of in-class vectors close to the cluster and the number of out-of-class vectors close to the cluster. These statistics are obtained from the training data; we simply count the number of training vectors that are near each cluster. Let $hist_C^I[i]$ be the count of in-class parent vectors which are near in-class cluster i. Let $hist_C^O[i]$ be the count of out-of-class parent vectors which are near in-class cluster i. These numbers bias the coin such that it lands with a vote for in-class with probability

$$P^I[i] = \frac{hist_C^I[i]}{hist_C^I[i] + hist_C^O[i]} \tag{5.3}$$

and lands with a vote for out-of-class with probability $1 - P[i]^I$.

Continuing the analogy, we can flip another biased coin whose probabilities depend on the out-of-class cluster histograms:

$$P^O[i] = \frac{hist_{\bar{C}}^O[i]}{hist_{\bar{C}}^I[i] + hist_{\bar{C}}^O[i]} \tag{5.4}$$

We see that the bias of the in-class coin is the in-class response of the cluster normalized by the sum of the in-class and out-of-class response. An analogous relationship holds for the out-of-class coin. Since these trials are independent, the total probability of the image belonging to the target class can be expressed as the sum of the log likelihood expressions. Let $H^C[i]$ be the number of test vectors near in-class cluster $C^I[i]$ and $H^{\bar{C}}[i]$ be the number of test vector near out-of-class cluster $\bar{C}^I[i]$, then the probability that the test vector is an in-class vector is

$$P(C) = \Sigma_{i=1}^{M} H^C[i] \log(P^I[i]) - \Sigma_{i=1}^{\bar{M}} H^{\bar{C}}[i] \log(P^O[i]) \tag{5.5}$$

where M is the number of in-class clusters and $\bar{M}$ is the number of out-of-class clusters. If $P(C) > t$ for some threshold t then the image is classified as belonging to C, otherwise it is classified as $\bar{C}$.

The Bernoulli idea is an approximation to the Bayes' rule computation using the Gaussian mixture model. The cluster histograms in equations 5.3 and 5.4 approximate the Bayes formula in equation 5.1 using a set of discrete observations.

This evaluation procedure is much cheaper to compute at run-time than the mixture of Gaussians. It is a scalar sum of logarithms, and does not require the Bayes rule computation. Experiments indicate that the Bernoulli trial method is an order of magnitude faster than evaluating the mixture model on typical data sets. Classification and speed performance are presented in detail in Chapter 6.

5.3 Discussion

There are many possible statistics for evaluating a test image. The mixture of Gaussian model and Bernoulli trials are two implementations. The mixture model is well-characterized since it is a smooth, parameterized distrubtion that can be evaluated using Bayes rule in a principled way. It is expensive to compute however, and does not actually model the true distribution of parent vectors in the target class. Results in Chapter 6, however, demonstrate the good performance of the mixture model.

The Bernoulli evaluation procedure is much faster to compute, but relies on less formal reasoning. It also requires extra training data in the form of cluster response histograms. Chapter 6 will show that its performance is also very good, though slightly behind the mixture model.

Both evaluation procedures, however, make the inaccurate assumption that parent vectors occur independently. This assumption may simplify the probability calculations, but probably decreases classification performance. Parent vectors from a test image form trees with implicit dependencies between vectors, and in a formal sense should not be evaluated independently.

On the other hand, there is a benefit to evaluating parent vectors independently. Adjacent parent vectors will share values in their low-resolution components because of their tree-like structure in the pyramid transform. We can then analyze vectors starting with lowest resolution components, and reuse the calculations when we encounter adjacent vectors which overlap with the first vector we operated upon. We take advantage of this property in our distance function **near()**, which compares parent vectors starting with the lowest resolution components. If the lowest resolution components are not with threshold, we do not have to check the higher frequency components of the vector.

Chapter 6

Results

This chapter presents results using the statistical model from Chapter 4 and the two evaluation procedures from Chapter 5. The experimental data included images of two target objects, human faces and cars, and background images randomly selected from the World Wide Web. In the experiments, we took a group of the target object images and computed their probability in the model. Then, we computed the probabilities of a large number of background images using the same model with the same parameters. We saw that the probabilities for the target class images were consistently higher than the probabilities for the background class images.

Because the probabilities are a continuum of floating-point values, we need to set a threshold for determining a test image's class membership. We show a series of receiver operating characteristic graphs which illustrate the detection and misclassification trade-offs associated with a particular threshold. The system is flexible in that the threshold can easily be moved to satisfy different performance criteria.

6.1 Face Detection

The majority of our experiments involved the face test set. We think this is an interesting domain for several reasons. As we noted in Chapter 1, some features of a face have similar appearance through pose and illumination changes, while other features are deformed or occluded. Currently there is no general-purpose face detection

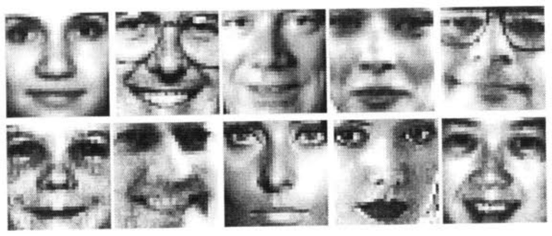

Figure 6-1: Example face images from the training set.

strategy which robustly handles these types of changes. Our model, on the other hand, takes advantage textural properties and allows for features in the image to be rearranged or missing. We show that our model built from frontal face views has good performance detecting non-frontal faces as well.

6.1.1 Building the face model

We have built a model of faces from a set of 1060 face images cropped from photographs found on the World Wide Web. The faces were chosen to be approximately frontal. There were small variations in scale as well as large variations in lighting and image quality. Each face image was scaled to be 32×32 pixels, converted to gray scale and then histogram equalized to reduce the variations from lighting. Some example face images are shown in figure 6-1 (shown before histogram equalization).

To build a background model of non-face images, we selected windows of random size and position from a set of Web images which did not contain people in them. The windows were then scaled to be 32×32 pixels and histogram equalized as with the faces. We used 1016 such non-face examples.

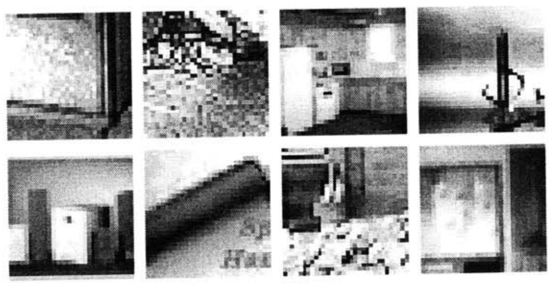

Figure 6-2: Example background images from the training set.

The parent vectors for each of the example faces and non-faces were computed as described in section 3.2.2. This yielded 1,085,440 (= $1060 \times 32 \times 32$) face parent vectors and 1,040,384 (= $1016 \times 32 \times 32$) non-face parent vectors. We used parent vectors with 4 scales and 6 oriented edge filters per scale. Including the high and low pass residuals, this resulted in parent vectors with 26 components.

To construct face and non-face models we first applied the clustering algorithm described in section 4.2 to the two sets of parent vectors separately. For the face model we used a threshold of 0.7 for all T_i in the function **near()**. For non-faces we used a threshold of $T_i = 0.8$ for all i. As also discussed in section 4.2, these thresholds were motivated from our experience with texture synthesis which used the same **near()** function. Although the synthesis results provided us with a good starting point for the thresholds, ultimately the values $T_i = 0.7$ and $T_i = 0.8$ were chosen since they gave the best end-to-end classification results. In comparison, the best synthesis results used values in the range $T_i = 0.1$ to 0.4. Thus, setting the thresholds still requires some trial and error experimentation. The clustering algorithm yielded 351,615 different clusters for the face parent vectors and 155,222 different clusters for the non-face parent vectors.

Next we applied discriminative analysis as discussed in section 4.3 to reduce the number of clusters and improve the models. This analysis yielded a face model with 1447 clusters and a non-face model with 483 clusters. A mixture of Gaussian model was built from these clusters as described in section 4.2. With these numbers of clusters, evaluating a 32×32 pixel image requires approximately 3 seconds on a Pentium II 300 mHz machine.

6.1.2 Correlating clusters with perceptual features

We can improve our intuition for the cluster-based model by verifying that some clusters correspond to particular perceptual features in a face. In other words, does a cluster encode a feature in the image such as an eye or lip that is useful for face detection? To test this hypothesis, we collected a new set of 266 face images and computed the parent vectors for these images. We then selected a parent vector

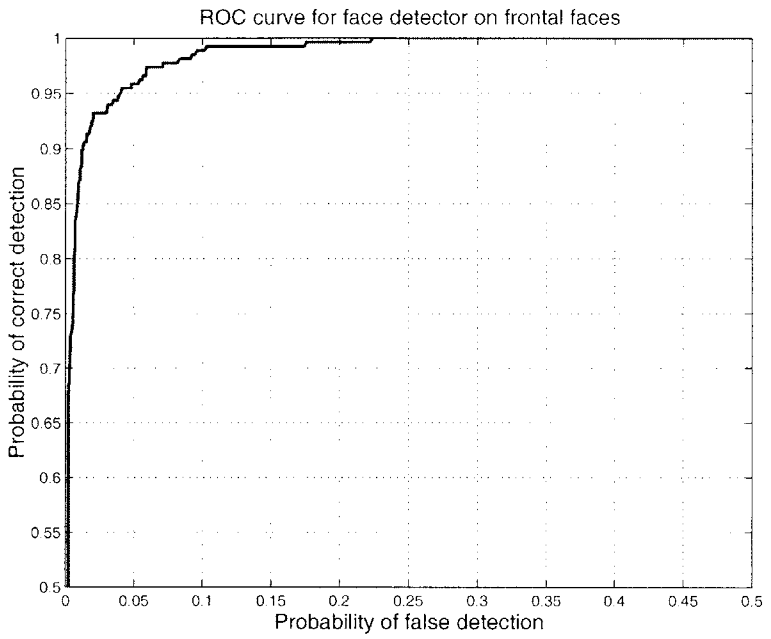

Figure 6-3: Receiver operating characteristics curve for a test set of 266 frontal faces and 2500 non-faces using the mixture of Gaussians classifier. Note that to focus on the interesting part of the graph, the correct detection rate axis begins at 50%.

cluster from the model and highlighted test image regions containing parent vectors from within this cluster. Cluster membership was determined using the distance function and thresholds defined before. Figure 6-4 shows the location where a "lip cluster" responds in several face images. While this cluster only responds to 6% of the *test* faces, in almost every case the response is localized near the lip region. In each of the remaining faces their is no response. Although this is a small percentage of faces, together the 1447 clusters provide enough coverage to detect most faces. Figure 6-5 shows the responses for two other clusters. The eye-corner cluster responds to approximately 30% of the faces, while the cheek cluster responds to about 4% of the faces.

6.1.3 Face classification results

To test the model we collected a new set of face and non-face images from the Web and preprocessed them as before. We used the same 266 face images from section 6.1.2 and 2500 non-face images in the test set. For each image, the parent vectors

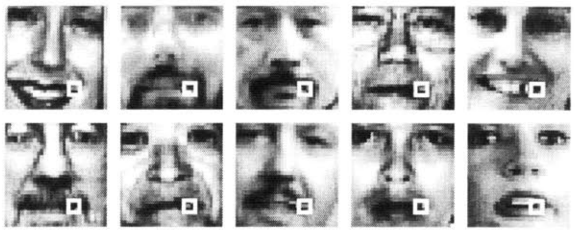

Figure 6-4: A cluster was chosen from the face model and compared to all parent vectors in each test image. The white boxes show the postion in each image where a parent vector was near the cluster. This cluster apparently represents a lip-corner feature.

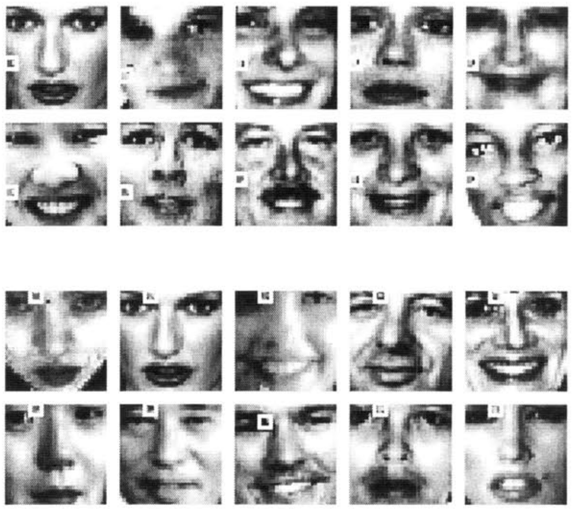

Figure 6-5: Results for two more face clusters. Again, the white boxes show the position in each image where a parent vector was near the cluster. The top set of images shows responses for a cheek cluster while the bottom set shows responses for an eye-corner cluster.

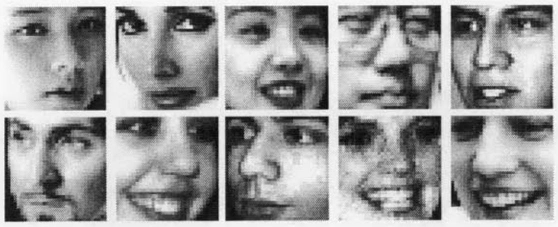

Figure 6-6: Example non-frontal face images which the face detector correctly classified.

were computed and then the test in equation 5.2 was applied to classify the image.

The results are shown as a receiver operating characteristics curve (ROC curve) in figure 6-3. The ROC curve shows promising preliminary results. For example, we get 90% correct detection with a false positive rate of 1.2% or 70% correct detection with 0.28% false positives.

The above ROC curve demonstrates performance on independently selected patches. There is no redundancy in this dataset. We believe that when placed in an end-to-end system this detector will demonstrate improved detection and rejection rates. One obvious refinement is to let the system bootstrap its non-face training set by adding false positive images into the non-face training set and relearning the model.

Classifying non-frontal faces

The main advantage of our framework for object detection is that it should be able to handle more variability than most other classifier-based methods. With faces this means that we expect parent vectors for frontal face images to be similar to parent vectors for non-frontal faces. To test this, we evaluated our face detector on a set of non-frontal faces, some of which are shown in figure 6-6. We used the same face detector just described which was trained on frontal faces. The preliminary results support the robustness of our framework. The ROC curve for a test set of 118 non-frontal faces and the same set of 2500 non-faces is shown in figure 6-7. The results are surprisingly good considering that non-frontal faces were not used in the training set. For example we get a correct detection rate of 80% with a false positive rate of 5% or a correct detection rate of 60% with a false positive rate of 1.2%. These results

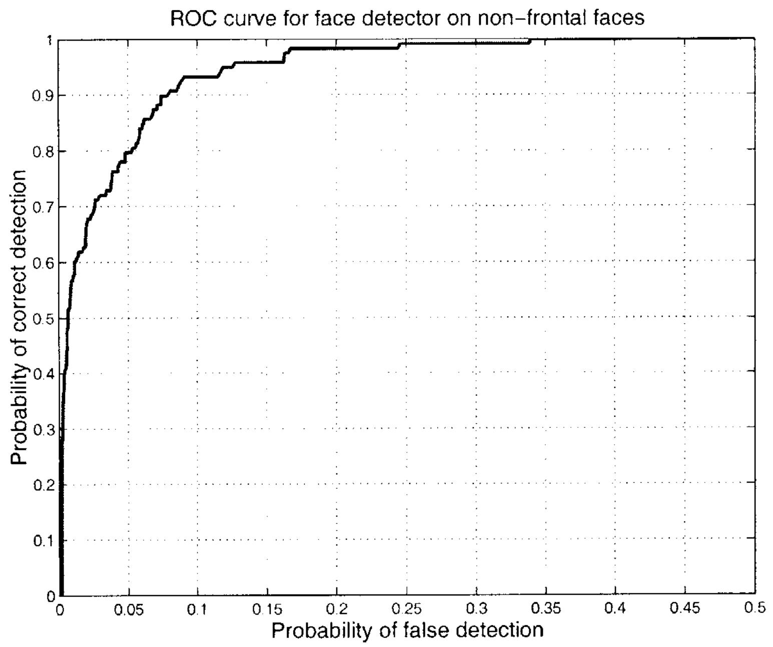

Figure 6-7: Receiver operating characteristics curve for a test set of 266 non-frontal faces and 2500 non-faces.

should be improved by including non-frontal faces in the training set.

6.1.4 Bernoulli trial results

In addition to the Gaussian mixture model classification procedure, we evaluated face test images using Bernoulli trials outlined in section 5.2. We used the same 266 face images and 2500 non-face images as in the mixture model trials. For each image, the parent vectors were computed and then the test in equation 5.5 was applied to classify the image. The same 118 non-frontal faces were used in the non-frontal trials. Evaluating one 32×32 image takes approximately 0.3 seconds on Pentium II 300 mHz machine. This is about an order of magnitude improvement over the Gaussian evaluation time.

6.2 Car detection

As a second example, we also learned a model for side views of cars. We only had a database of 48 car images available for this experiment. However, since all the

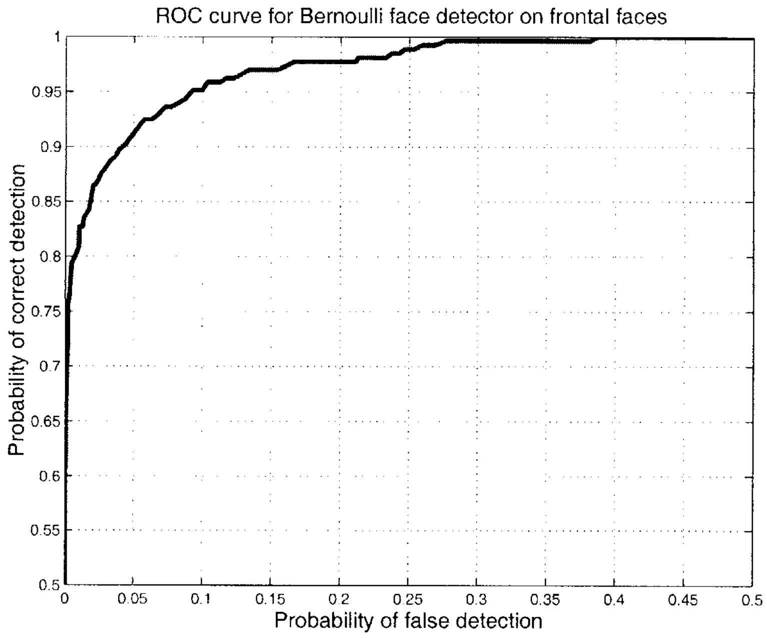

Figure 6-8: Receiver operating characteristics curve for a test set of 266 frontal faces and 2500 non-faces using the Bernoulli classifier. Note that to focus on the interesting part of the graph, the correct detection rate axis begins at 50%.

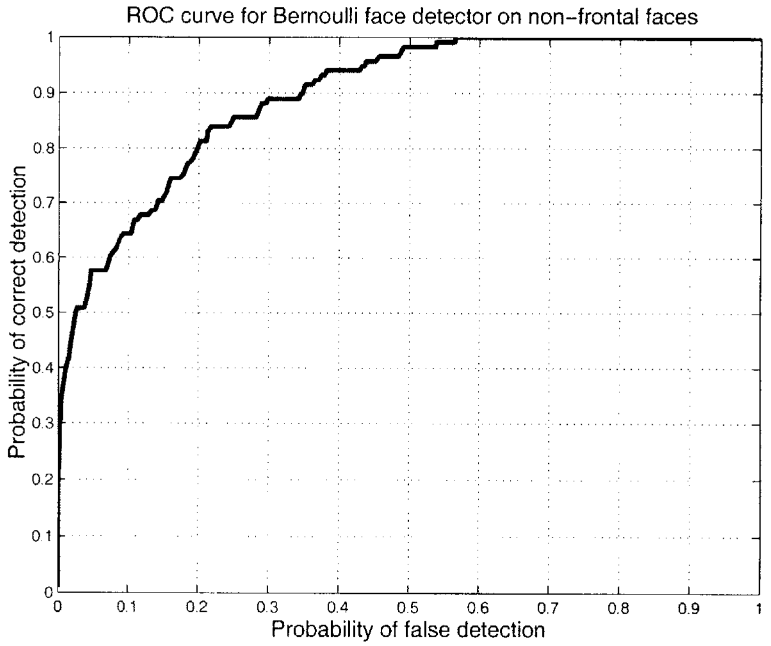

Figure 6-9: Receiver operating characteristics curve for a test set of 118 non-frontal faces and 2500 non-faces using the Bernoulli classifier. The performance of the Bernoulli detector is much worse than the mixture model in the non-frontal case%.

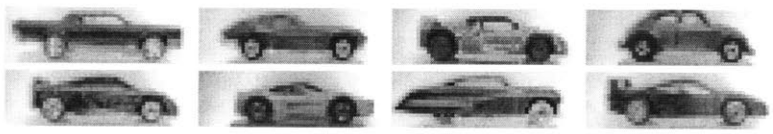
Figure 6-10: Eight of the 24 example cars used to train the car detector.

cars were photographed under similar conditions, our intuition was that this should be sufficient to learn a model. We split the set into 24 training images and 24 test images. We also used a set of 1024 non-car images to build a background model. The image size used was 52 pixels in width by 16 pixels in height. Figure 6-10 shows a few example cars. All of the car images were acquired by taking photographs of toy cars.

The parent vectors consisted of the values from 6 oriented edge filters computed over 3 scales. Including the high and low resolution residuals, the parent vectors contained 20 components each. Initial clustering of the car parent vectors yielded 12,295 clusters using a threshold of $T_i = 0.5$ in the distance function of equation 3.2. Initial clustering of the parent vectors from the 1024 non-car images yielded 32,405 clusters using a threshold of $T_i = 0.8$. After discriminative analysis, the car model contained 1229 clusters and the non-car model contained 1001 clusters.

This model was then tested on a test set of 24 car and 200 non-car images. The resulting ROC curve is shown in figure 6-11. The results are very good. Only 1 false positive was made with 100% correct detections.

6.3 Image database retrieval

It is difficult to compare the face detection results with those of Sung and Poggio or Rowley and Kanade. These systems process entire images; first decomposing them into a set of overlapping patches at multiple scales, and then classifying each patch. There are ten's of thousands of such patches in each image, most containing only background. While it is a daunting task to reject each and every background patch, it is important to point out that since they overlap, many of the patches are highly redundant. A patch that is correctly rejected because it does not contain a face is

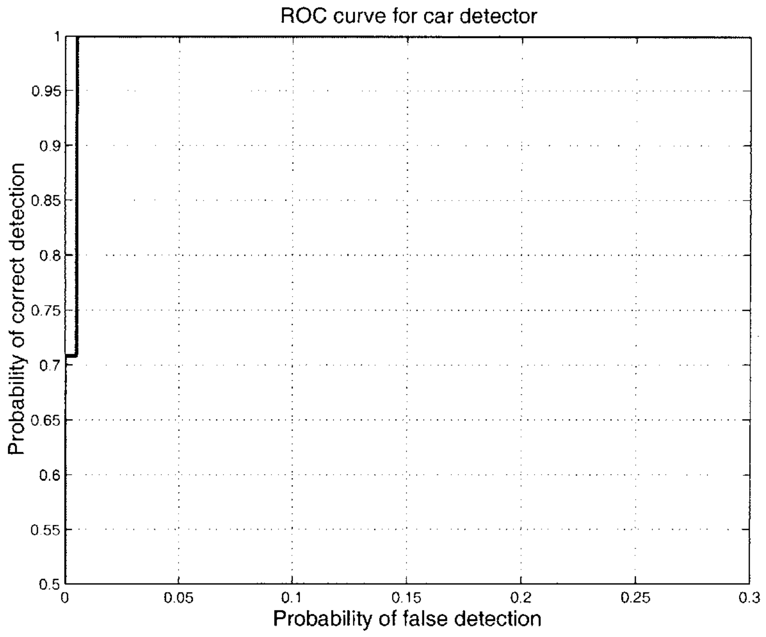

Figure 6-11: Receiver operating characteristics curve for a test set of 24 side views of cars and 200 non-cars.

very likely to be rejected if it shifted by a single pixel. A similar issue arises in the detection of faces. Since a single face will appear in many overlapping patches, there will be many opportunities to detect it.

While a complete system for detecting objects in uncropped images is still under development, at this point the computational complexity can be estimated. The first step is to form the image pyramid using convolution and downsampling. This requires roughly 1000 operations per pixel. After that, each parent vector of the image must be compared to the clusters in the face and non-face models. In most experiments there are 2000 clusters and 26 dimensions in a parent vector. A naive algorithm would require $2000 \times 26 = 52,000$ operations per pixel. Recall however that the image parent vectors are arranged in a tree such that comparisons at low resolutions levels can be reused at higher resolutions. By clever bookkeeping this can be reduced to $2000 \times 6 = 12,000$ operations per pixel. Finally, estimates from pixels in a region must be aggregated into a score for an entire patch. Once again by clever bookkeeping, this should require just a few operations per pixel. In the final analysis for a given scale, searching every overlapping patch requires something like 15,000

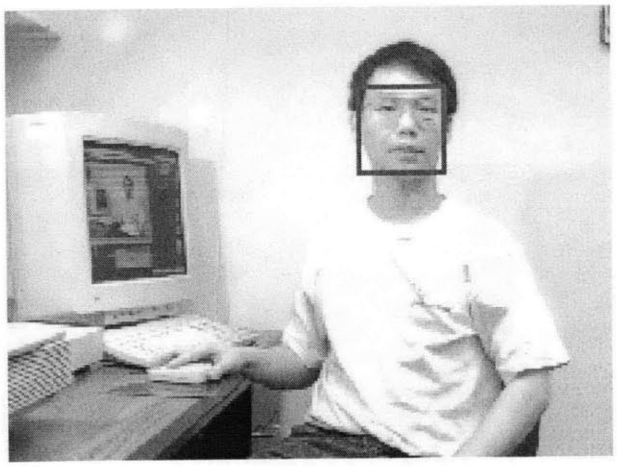

Figure 6-12: An example detection by the preliminary system

.

operations per pixel – or about 10^9 operations for a 256×256 pixel image. With additional refinement, our goal is to scan an image in seconds.

We have implemented a prelimary system for scanning entire images. See figures 6-12, 6-13, and 6-14 for several detection examples.

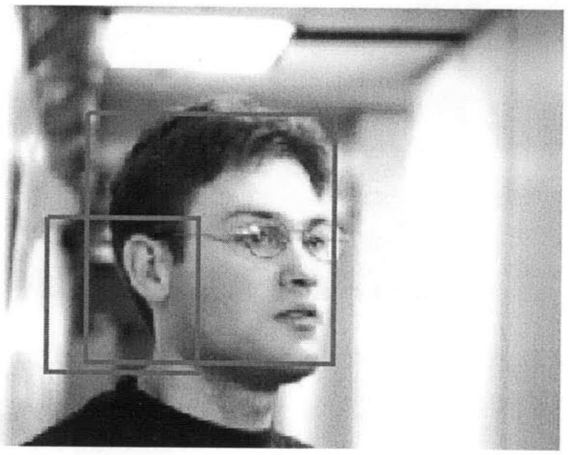

Figure 6-13: Notice that the center of the face has been detected, as well as a second region centered on the hair and ear. We suspect the that the training data contained enough examples of hair and ears for the model to respond to these textures in a test image.

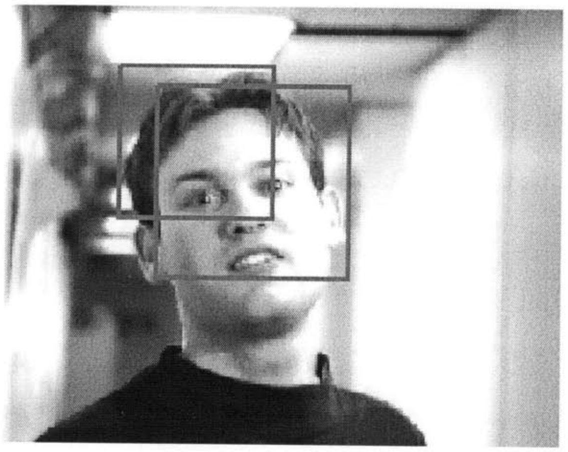

Figure 6-14: Another example where more than one patch of the face was found by the model.

Chapter 7

Conclusion

7.1 Summary

We have presented a new approach to detecting classes of objects which is based on De Bonet and Viola's recent technique for representing textures. The results we have obtained for face detection and car detection are very promising. They show that object detection for an object class with large variations can rely on texture recognition.

7.2 Future Work

Several interesting research questions arise from our current results. Our future work in this area will focus on implementing a face detector that searches over positions and scales. We also intend to improve the computational expense of the method to make it a practical solution for object detection. In particular, the following improvements will help improve the accuracy and efficiency of the model:

1. *Enable learning in the statistical model.* Currently the model is constructed from a large number of example images and does not modify cluster statistics as test data or new training examples are presented. When new data is classified correctly, statistics from the test images can be used to update the cluster means and variances or the histogram counts.

2. *Hierarchical search.* When performing a multi-scale search on a large image, a small, fast model could be used find candidate locations containing a concentration of signature parent vectors. Next, a larger more accurate model can be applied to these candidate locations. By chaining a series of increasingly complex models together, we can apply the most expensive model to only the most promising locations in the image.

3. *Use color information.* The first implementation of the model used only gray scale images. Building the model with color information should help in discrimination since it places additional constraints on parent vector values in the clustering and evaluation procedures.

Future experiments

In this section we suggest several specific experiments that will help us better understand the strengths and limitations of this statistical model. There are many open questions regarding this research.

1. *Build models using different multi-scale transforms.* The steerable wavelet pyramid used in this research is an overcomplete representation [18] that is expensive to compute. Since we do not require the transform to be invertible, we could use a simpler multi-scale transform. The Gaussian pyramid, for example, would be a good candidate since it can be computed very quickly.

2. *Histogram normalization through parent vector normalization.* If a large image is scanned for the target object by computing the probability of individual patches, it is unclear how to histogram equalize the patches of the image. One solution is to normalize the coefficients in the parent values directly instead of normalizing the original intensity values in the image.

3. *Rotation-invariance through parent vector normalization.* Robustness to rotation in the plane is a difficult problem. When the image rotates, the orientation

of the lines and ridges changes, and this redistributes the energy in the subbands of a multi-orientation transform. The total energy is preserved since no new features are created. However, the components of the parent vectors will be reordered.

If training images of target class happens to have a handful of trademark parent vectors that consistently appear in a particular orientation, then the statistical model will find a cluster of vectors with similar values associated with a particular orientation. When a novel image is encountered that has these same trademark values appearing at a different orientation subband, then we could postulate that the novel image is just an instance of the target class which has been rotated. We can use this information to rotate the test image back to the canonical orientation the model was trained on.

This technique builds in rotation-invariance to the model and requires a minimal amount of computation compared to other techniques for rotation-invariance [2].

4. *Scale-invariance through parent vector normalization.* We may be able to include scale invariance using an idea similar to the rotation-invariance normalization. In this case, however, we look for a few trademark values appearing in a scale subband. If a novel test image has the trademark values appearing in a different scale subband, then we can use this information to rescale the image so that the trademark values are moved into the scale subband the model learned during training.

7.3 Discussion

In addition to these experimental issues, there are number of parameters which we chose during this research which may not be optimal. We used a fixed model size of 32 x 32 pixels. A lower resolution model may have better performance because the high-frequency details could be overwhelmed by noise artifacts. Moreover, there are fewer scales in the multi-scale transform so the parent vectors will be of lower dimensionality.

Evaluating a test may be faster in some cases because of the reduced dimensionality. On the other hand, using higher resolution images may improve performance since more detail is available in the features to help discriminate between the target and the background.

There are several parameters in the clustering algorithm which are also *ad hoc*. The choices for the distance function and parameters are still not formally justified.

We have trained a model on frontal faces and tested it on non-frontal faces with good results. If we were to train a new model using both frontal and non-frontal examples, we would hope that the non-frontal images would provide more examples of ears, hair, or other features that are not as visible in the frontal images. These features would become part of the library of textures in the model in addition to the textures extracted from frontal views. We would imagine that the overlap in features between the frontal and non-frontal faces would reinforce the model and improve detection rates for both views.

This thesis has demonstrated the usefulness of texture in object detection. It is a flexible method robust to the rearrangement of features, occlusion, and variation in lighting. There are many possibilities for improving this general model for application in a wide range of detection problems. It has the potential to solve several hard computer vision problems with the principled tools of statistics.

Bibliography

[1] D. Beymer, A. Shashua and T. Poggio, "Example Based Image Analysis and Synthesis", *A.I. Memo 1431*, MIT, 1993.

[2] M.C. Burl, T.K. Leung, P. Perona, "Face Localization via Shape Statistics", in *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 1995, pp. 154-159.

[3] J. De Bonet, "Multiresolution sampling procedure for analysis and synthesis of texture images," in *Computer Graphics.* ACM SIGGRAPH, 1998.

[4] J. De Bonet and P. Viola, "Texture recognition using a non-parametric multi-scale statistical model," *In IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[5] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973, pp. 85-89.

[6] G.J. Edwards, C.J. Taylor and T.F. Cootes, "Interpreting Face Images using Active Appearance Models" in *Proceedings of the 3rd International Conference on Automatic Face and Gesture Recognition*, IEEE, 1998, pp. 300-305.

[7] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, Inc., San Diego, CA. 1990.

[8] D. Heeger and J. Bergen, "Pyramid-based texture analysis/synthesis," In *Proceedings of ACM SIGGRAPH*, August 1995, pp 229-238.

[9] M. Jones and T. Poggio, "Multidimensional Morphable Models: A Framework for Representing and Matching Object Classes" in *International Journal of Computer Vision*, Volume 29, No. 2, August 1998, pp. 107-131.

[10] M. Lades, C.C. Vorbruggen, J. Buhmann, J. Lange, C. von der Malsburg, R.P. Wurtzand W. Konen, "Distortion Invariant Object Recognition in the Dynamic Link Architecture", *IEEE Transactions on Computers*, Vol 42, No 3, March 1993, pp. 300-311.

[11] B. Moghaddam, C. Nastar and A. Pentland, "Bayesian Face Recognition using Deformable Intensity Surfaces" in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

[12] C. Papageorgiou, M. Oren and T. Poggio, "A General Framework for Object Detection," in *Proceedings of the 6th International Conference on Computer Vision*, January 1998, pp 555-562.

[13] S. Pinker. *How the Mind Works.* John Wiley and Sons, 1997, pp. 85-89.

[14] K. Popat and R. Picard, "Cluster-based probability model and its application to image and texture processing," *IEEE Transactions on Image Processing*, Vol. 6, No. 2, Feb 1997, pp. 268-284.

[15] H. Rowley, S. Baluja, T. Kanade, "Human Face Detection in Visual Scenes," *Technical Report CMU-CS-95-158R*, CMU, 1995.

[16] B. Schiele and J. Crowley, "Recognition without Correspondence using Multidimensional Receptive Field Histograms" *M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 453*, December 1997.

[17] H. Schneiderman and T. Kanade, "Probabilistic Modeling of Local Appearance and Spatial Relationships for Object Recognition", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1998, pp. 45-51.

[18] E. Simoncelli and W. Freeman, "The Steerable Pyramid: A flexible architecture for multi-scale derivative computation," In *Int'l Conference on Image Processing*, Vol III, pp. 444-447, Washington, D.C., October 1995.

[19] E. Simoncelli and J. Portilla, "Texture Characterization via Joint Statistics of Wavelet Coefficient Magnitudes," in *Proceedings of the Fifth International Conference on Image Processing*, 1998.

[20] E. Simoncelli and R. Buccigrossi, "Embedded Wavelet Image Compression Based on a Joint Probability Model," in *IEEE International Conference on Image Processing*, 1997.

[21] K. Sung and T. Poggio, "Example-based learning for view-based human face detection", *A.I. Memo 1521*, MIT, December 1994.

[22] M. Turk and A. Pentland, "Face Recognition Using Eigenfaces" in *IEEE Conference on Computer Vision and Pattern Recognition*, 1991, pp. 586-591.

[23] P. Viola, "Complex Feature Recognition: A Bayesian Approach for Learning to Recognize Objects", *A.I. Memo 1591*, MIT, November, 1996.

[24] L. Wiskott, J.M. Fellous, N. Kruger and C. von der Malsburg, "Face Recognition by Elastic Bunch Graph Matching", *International Conference on Image Processing*, Vol. I, 1997.

[25] S. Zhu, Y. Wu and D. Mumford, "Filters, Random Fields and Maximum Entropy (FRAME) - Towards A Unified Theory For Texture Modeling" in *International Journal of Computer Vision*, Vol 27, No 2, March 1998, pp. 107-126.