HAND GESTURE RECOGNITION AND FACE DETECTION IN IMAGES

BY

MING-HSUAN YANG

B.S., National Tsing-Hua University, 1991
M.S., University of Southern California, 1992
M.S., University of Texas at Austin, 1994

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2000

Urbana, Illinois

Acknowledgments

I would like to express my gratitude to my advisor, Professor Narendra Ahuja. My graduate career could not have been an enriching and enjoyable as it was it it were not for his guidance and support. His insightful comments has been an inspiration for my work. It has been a great pleasure to have Professor Ahuja as my advisor.

I am grateful to Professor Dan Roth and Professor David Kriegman for their guidance. From them, I have learned how to conduct quality research in different research subjects. I also thank Prof. Thomas Huang for his insightful comments and suggestions on my thesis work.

I thank my parents and my wife, Yi-Ming, for their love and support that kept me pursuing and focusing on my graduate study. I would not have accomplished this thesis if there were not their support.

I have had valuable conservations with my colleagues during the course of my research. I would like to thank all the members in my lab.

This thesis is dedicated to my parents and my wife for their love and support.

Table of Contents

Chapter 1 Introduction **1**
1.1 Gesture Recognition 2
1.2 Face Detection 3
1.3 Learning To Recognize 3D Objects 5
1.4 Geometric Approach to Train Support Vectors Machines 7
1.5 Thesis Overview 9

Chapter 2 Literature Review on Face Detection Methods **11**
2.1 Knowledge-Based Top-Down Methods 15
2.2 Bottom-Up Feature-Based Methods 18
2.2.1 Facial Features 19
2.2.2 Texture 22
2.2.3 Skin Color 23
2.2.4 Multiple Features 25
2.3 Template Matching 28
2.3.1 Simple Templates 28
2.3.2 Deformable Templates 29
2.4 Appearance-Based Methods 31
2.4.1 Eigenfaces 32
2.4.2 Distribution-Based Methods 33
2.4.3 Neural Networks 36
2.4.4 Support Vector Machines 39
2.4.5 Sparse Network of Winnows 40
2.4.6 Bayesian Approach 40
2.4.7 Hidden Markov Model 41
2.4.8 Information-Theoretical Approach 44
2.4.9 Inductive Learning 44
2.5 Face Image Databases and Performance Evaluation 45
2.5.1 Face Image Database 46
2.5.2 Benchmark Test Sets for Face Detection 47
2.5.3 Performance Evaluation 51
2.6 Discussion and Conclusion 53

Chapter 3 Recognizing Hand Gestures Using Motion Trajectories **55**
3.1 Introduction 55
3.2 Related Work 56
3.3 Motion Segmentation 58

3.3.1 Multiscale Image Segmentation . . . 58
3.3.2 Region Matching . . . 59
3.3.3 Affine Transformation Estimation . . . 60
3.3.4 Motion Field Integration . . . 61
3.3.5 Motion Field Segmentation . . . 61
3.4 Color and Geometric Analysis . . . 64
3.5 Motion Trajectories . . . 66
3.6 Motion Pattern Classification . . . 69
3.7 Experiments . . . 70
3.8 Discussion and Conclusion . . . 70

Chapter 4 Skin Color Model . . . 72
4.1 Proposed Mixture Model . . . 73
4.1.1 Gaussian Mixture Model . . . 73
4.1.2 Estimating Parameters Using EM Algorithm . . . 74
4.2 Statistical Tests . . . 75
4.2.1 Hawkins' Test for Normality and Homoscedasticity . . . 75
4.2.2 Statistical Test for the Number of Components . . . 78
4.3 Experimental Results . . . 80
4.3.1 Estimated Density Function . . . 80
4.3.2 Hawkins' Test on Normality and Homoscedasticity . . . 82
4.3.3 Bootstrap Test on the Number of Components . . . 82
4.4 Applications . . . 83
4.5 Discussion and Conclusion . . . 84

Chapter 5 Face Detection . . . 86
5.1 Introduction . . . 86
5.2 Detecting Faces in Color Images . . . 88
5.2.1 Multiscale Segmentation . . . 88
5.2.2 Human Skin Color Model . . . 90
5.2.3 Geometric Analysis and Postprocessing . . . 90
5.2.4 Experimental Results . . . 91
5.2.5 Conclusion . . . 93
5.3 Mixture of Factor Analyzers . . . 93
5.3.1 Factor Analysis . . . 93
5.3.2 Mixture Model . . . 94
5.3.3 Detecting Face Patterns . . . 96
5.4 Mixture of Linear Spaces Using Fisher Linear Discriminant . . . 96
5.4.1 Labeling Samples Using Self-Organizing Map . . . 96
5.4.2 Fisher Linear Discriminant . . . 97
5.4.3 Class-Conditional Density Function . . . 98
5.4.4 Detecting Face Patterns . . . 99
5.5 Experiments . . . 99
5.6 Discussion and Conclusion . . . 102

Chapter 6 Learning To Recognize 3D Object With SNoW 103
6.1 Introduction 103
6.2 Learning Framework 105
6.2.1 Learning Scenario 106
6.2.2 Robustness 109
6.3 From Theory to Practice 109
6.4 The SNoW Architecture 111
6.5 View-Based Methods 115
6.5.1 Support Vector Machines 116
6.6 Experimental Evaluation 118
6.6.1 Ground Truth of the COIL-100 Dataset 119
6.6.2 Experiment Setups 120
6.6.3 Results Using Pixel-Based Representation 121
6.6.4 Results Using Edge-Based Representation 122
6.6.5 Noise Model 123
6.7 Discussion and Conclusion 124

Chapter 7 Geometric Approach To Train Support Vector Machines 125
7.1 Introduction 125
7.2 Related Work 128
7.3 Extracting Guard Vectors for Support Vector Machines 129
7.3.1 Determining Guard Vectors Using Linear Programming 129
7.3.2 Linear SVM 134
7.3.3 Nonlinear SVM 135
7.4 Experiments 136
7.4.1 Synthetic Data 136
7.4.2 Real-World Data 137
7.5 Discussion and Conclusion 139

Chapter 8 Conclusion and Future Work 141
8.1 Conclusion 141
8.2 Future Work 142

References 144

Vita 167

List of Tables

2.1 A categorization of methods for face detection in still images 15
2.2 Face image database 48
2.3 Test sets for face detection 49
2.4 Experimental results on images from test set 1 (125 images with 483 faces) and test set 2 (23 images with 136 faces) (see text for details) 52

4.1 Results of Hawkins' test for normality and homoscedasticity(applied to data in known classified form) 82

5.1 Experimental results on images from test set 1 (125 images with 483 faces) in [161] . 100
5.2 Experimental results on images from test set B (20 images with 136 faces) in [188] . 100

6.1 Recognition rates of nearest neighbor classifier 120
6.2 Experimental results of three classifiers using the 100 objects in the COIL-100 dataset 121
6.3 Recognition rates of SNoW using two learning paradigms 122
6.4 Experimental results of three classifiers using the 100 objects in the COIL-100 dataset 123
6.5 Experimental results of SNoW classifier on occluded images with 36 views per object 124

7.1 Experimental results on synthetic data: Each data set has m points that are randomly scattered in an n-dimensional hypercube. 138
7.2 Experimental results on benchmark data sets: The first one is the Wisconsin Breast Cancer data set in which each of the 699 points has 9 features. The second data set consists of 500 20×20 face images and 1000 20×20 nonface images for face detection. 138

List of Figures

2.1 Original and mosaic images at different resolutions. Each square cell consists of $n \times n$ pixels in which the intensity of each pixel is replaced by the average intensity of the pixels in that cell. As n increases, the resolution of cell is lower and contains less detail. Therefore, an algorithm needs to search fewer pixels in a low resolution mosaic image to locate the face. 16

2.2 A typical face used in knowledge-based top-down methods: Rules are coded based on human knowledge about the characteristics (e.g., intensity distribution and difference) of the facial regions. 17

2.3 Horizontal and vertical profiles of mosaic images. It is feasible to detect a single face by searching for the peaks in horizontal and vertical profiles. However, the same method has difficulty to detect faces in complex backgrounds or multiple faces as shown in 2.3(b) and 2.3(c). 18

2.4 Face and nonface clusters used by Sung and Poggio. Their method estimates density functions for face and nonface patterns using a set of Gaussians. The estimated distributions are then used to compute the Mahalanobis distances from each image pattern to the cluster centers. These distances are then used for detection. 34

2.5 Prototype of each face class using Kohonen's SOM by Yang, Ahuja and Kriegman. Each prototype corresponds to the center of a cluster. 35

2.6 System diagram of Rowley's method. Each face is pre-processed before feeding it to an ensemble of neural networks. Several arbitration methods are used to determine whether a face exists based on the output of these networks. 39

2.7 Hidden Markov Model for face localization. (a) Observation vectors: To train an HMM, each face sample is converted to a sequence of observation vectors. Observation vectors are constructed from a window of $W \times L$ pixels. By scanning the window vertically with P pixels of overlap, an observation sequence is constructed. (b) Hidden states: When an HMM with five states is trained with sequences of observation vectors, the boundaries between states are shown in (b). 42

2.8 Sample images in Sung's dataset. Some of these images are scanned from newspapers and thus have low resolution. Though most faces in the images are upright and frontal. Some faces in the images appear in different pose. 49

2.9 Sample images in Rowley's dataset. Some images contain hand-drawn cartoon faces. Most images contain more than one faces. Also, the face size varies. 50

2.10 Sample images of non-upright faces from Rowley's dataset. This dataset contains faces in different orientation and some faces in different pose. 51

2.11 Different criteria lead to different detection results. Suppose all the subimages in (b) are classified as face patterns by a classifier. A loose criterion may declare all the faces as "successful" detects while a more strict one would declare most of them as nonfaces. 53

3.1 Image sequence of ASL sign "cheerleader" 62
3.2 Motion segmentation of the image sequence "cheerleader" (pixels of the same motion region are displayed with same gray level and different regions are displayed with different gray levels) 63
3.3 Extracted head and palm regions from image sequence "cheerleader" 63
3.4 Extracted gestural motion trajectories from segments of ASL sign "cheerleader" (since all pixel trajectories are shown, they form a thick blob) 64
3.5 Image sequence of ASL sign "any" (time increases left to right and top to bottom) . 64
3.6 Motion segmentation of the sequence in Figure 3.5 (time increases left to right and top to bottom) 65
3.7 Extracted human head and palm regions in the sequence of Figure 3.5 65
3.8 Image sequence of ASL sign "anything" (time increases left to right and top to bottom) 65
3.9 Extracted gestural motion trajectories (subsampled by a factor of 10) of ASL sign "cheerleader" 67
3.10 Motion patterns of gesture "any" 68
3.11 Motion patterns of gesture "anything" 68
3.12 Architecture of TDNN 69

4.1 Histogram of skin color (downsampled by a factor of 10) viewed from different angles 81
4.2 Estimated Density Function 81
4.3 Original image and results of skin detection 83
4.4 Image sequence of ASL sign "any" (time increases left to right and top to bottom) . 84
4.5 Detected skin areas in the sequence in Figure 4.4 84

5.1 Results of multiscale segmentation 90
5.2 Intermediate and final results of face detection 91
5.3 Faces of different size and orientation 92
5.4 Faces with different features 92
5.5 Human faces in complex background 92
5.6 Prototype of each face class 97
5.7 Sample experimental results using mixture of factor analyzers on images from three test sets. Every detected face is shown with an enclosing window. 101
5.8 Sample experimental results using mixture of subspaces with Fisher Linear Discriminant on images from three test sets. Every detected face is shown with an enclosing window. 101

6.1 SNoW: *Objects and Notation* 113
6.2 SNoW: *Training and Evaluation.* Training is the learning phase in which the network is constructed and weights are adjusted. Evaluation is the phase in which the network is evaluated, given an observation. This is a conceptual distinction; in principle, one can run in on line mode, in which training is done continuously, even when the network is used for evaluating examples. 113
6.3 SNoW: *Main Procedures* 114

6.4 Columbia Object Image Library (COIL-100) consists of 100 objects of varying poses 5° apart). The objects are shown in row order where the highlighted ones are those considered more difficult to recognize in [147]. 118
6.5 Mismatched objects using the nearest neighbor method. $(x : a, y : b)$ means that object x with view angle a is recognized as object y with view angle b. It shows some of the 54 errors (out of 3,600 test samples) made by the nearest neighbor classifier when there are 36 views per object in the training set. 119
6.6 Objects images with and without occlusion as well their edge maps. 123

7.1 Dual representation of a point. Each point in the primal plane corresponds to a line in the dual plane. 130
7.2 Using linear programming to extract guard vectors. Each point and its dual are shown in the figure. Points p_1 and p_2 belong to class '+' while point p_3 belongs to class '-'. 131
7.3 Geometric interpretation of a linear programming problem. (a) shows one case where a feasible solution exists. (b) shows one case where no feasible solution exists. 132
7.4 Geometric interpretations of guard vectors and support vectors (guard vectors form a superset of support vectors): guard vector 1 and 4 are the support vectors for the optimal separating hyperplane found by the SVM for this set of points. 133
7.5 Benchmark results of the proposed algorithm against a linear SVM in terms of CPU time and memory requirements using the first 2-dimensional data set shown in Table 7.1. 137
7.6 Some 20×20 face images used in the experiment. Each image is converted to a 400-dimensional vector. 139

Chapter 1

Introduction

The goal of this thesis is to develop vision-based interfaces between man and machine. Various aspects of research on intelligent human computer interaction are addressed in the context of computer vision and machine learning.

The first part of this thesis aims to extract two-dimensional motion across image frames and classify underlying three-dimensional motion patterns. We have developed a method for extracting and classifying two-dimensional motion in an image sequence based on motion trajectories. For concreteness, we focus on image sequences showing hand motions for signs in American Sign Language. The same method can be adapted to recognize other motion patterns such human walking and running.

In this context, it is important to identify certain body parts of the signer in the image sequence. Also, it is important to locate the hand position relative to head and the rest of the body. Therefore, a related problem is to detect human faces robustly. In the second part of this thesis, three methods have been developed to detect human faces in color or gray-level images.

Learning has received increasing attention in recent years. In the third part of this thesis, we have developed a distribution free learning theory to an archetypical visual recognition problem: object recognition. The problem is viewed as that of learning a representation of an object that, given a new image, is used to recognize the target object in it. The learning account is developed within the PAC (Probably Approximately Correct) model of learnability [202]. For this framework to contribute to a practical solution, there needs to be a computational approach that is able to learn the concepts of objects. The evaluation we provide for this framework relies on the SNoW learning architecture [22] that is used in a large scale object recognition experiment.

Recently, Support Vector Machines (SVMs) have shown great potential in numerous visual learning and pattern recognition problems. However, training an SVM for a large-scale problem is challenging since it is computationally intensive and the memory requirement grows with square of the number of training vectors. In the fourth part of this thesis, we have developed a geometric approach to train SVMs and compared its performance against conventional methods.

We give overviews of our research in hand gesture recognition, face detection and learning to recognize objects in the following sections.

1.1 Gesture Recognition

Our research is concerned with extracting two-dimensional motion fields of objects across a video sequence and classifying each as one of a set of *a priori* known classes. The algorithm is used to recognize dynamic visual processes based on spatial, photometric and temporal characteristics. An application of the algorithm is in sign language recognition where an utterance is interpreted based on, for example, hand location, shape, and motion. The performance of the algorithm is evaluated on the task of recognizing 40 complex hand gestures of American Sign Language (ASL).

The algorithm consists of two major steps. First, each image is partitioned into regions using a multiscale segmentation method. Regions between consecutive frames are then matched to obtain 2-view correspondences. Affine transformations are computed from each pair of corresponding regions to define pixel matches. Pixel matches over consecutive image pairs are concatenated to obtain pixel-level motion trajectories across the video sequence. Pixels are also grouped based on their 2-view motion similarity to obtain a motion based segmentation of the video sequence. Only some of the moving regions correspond to visual phenomena of interest. Both the intrinsic properties of the objects represented by image regions and their dynamics represented by the motion trajectories determine whether they comprise an event of interest. For example, it is sufficient to recognize most gestures in ASL in terms of shape and location changes of palm regions. Therefore, palm and head regions are extracted out in each frame and the palm locations are specified with reference to the usually still head regions.

To recognize motion patterns from trajectories, we use a time-delay neural network (TDNN) [206]. TDNN is a multilayer feedforward network that uses time-delays between all layers to

represent temporal relationships between events in time. An input vector is organized as a temporal sequence, where only the portion of the input sequence within a time window is fed to the network at one time. The time window is shifted and another portion of the input sequence is given to the network until the whole sequence has been scanned through. The TDNN is trained using standard error backpropagation learning algorithm. The output of the network is computed by adding all of these scores over time, followed by applying a nonlinear function such as sigmoid function to the sum. TDNNs with two hidden layers using sliding input windows over time lead to a relatively small number of trainable parameters. We adopt TDNN to recognize motion patterns because gestures are spatio-temporal sequences of feature vectors defined along motion trajectories. Our experimental results show that motion patterns can be learned by a time-delay neural network with high recognition rate.

1.2 Face Detection

Images of human faces are central to intelligent human computer interaction. Much research is being done involving face images, including face recognition, face tracking, pose estimation, expression recognition and gesture recognition. However, most existing methods on these topics assume human faces in an image or an image sequence have been identified and localized. To build a fully automated system that extracts information from images of human faces, it is essential to develop robust and efficient algorithms to detect human faces. Given a single image or a sequence of images, the goal of face detection is to identify and locate all of the human faces regardless of their positions, scales, orientations, poses and lighting conditions. This is a challenging problem because human faces are highly non-rigid objects with a high degree of variability in size, shape, color and texture. Most recent methods for face detection can only detect upright, frontal faces under certain lighting conditions. In this thesis, we present one method to detect faces in color images and two methods that use mixtures of linear subspaces to detect faces with different features and expressions, in different poses, and under different lighting conditions.

The first method uses structure and color information to detect faces in color images. A human skin color model is built to capture the chromatic properties based on multivariate statistical analysis. Given a color image, multiscale segmentation is used to generate homogeneous regions at

multiple different scales. From the coarsest to the finest scale, regions of skin color are merged until the shape is approximately elliptic. Postprocessing is performed to determine whether a merged region contains a human face and include the facial features of non-skin color such as eyes and mouth if necessary. Experimental results show that human faces in color images can be detected regardless of size, orientation and viewpoint.

Since the images of a human face lie in a complex subset of the image space that is unlikely to be modeled by a single linear subspace, we use a mixture of linear subspaces to model the distribution of face and nonface patterns. The second method is an extension of factor analysis. Factor analysis (FA), a statistical method for modeling the covariance structure of high dimensional data using a small number of latent variables, has some analogies with principal component analysis (PCA). However PCA, unlike FA, does not define a proper density model for the data since the cost of coding a data point is equal anywhere along the principal component subspace (i.e. the density is unnormalized along these directions). Further, PCA is not robust to independent noise in the features of the data since the principal components maximize the variances of the input data, thereby retaining unwanted variations. Hinton et al. have applied FA to digit recognition and they compare the performance of PCA and FA models [72]. A mixture model of factor analyzers has recently been extended [58] and applied to face recognition [54]. Both studies show that FA performs better than PCA in digit and face recognition. Since pose, orientation, expression, and lighting affect the appearance of a human face, the distribution of faces in the image space can be better represented by a mixture of subspaces where each subspace captures certain characteristics of face appearances. We present a probabilistic method that uses a mixture of factor analyzers (MFA) to detect faces with wide variations. The parameters in the mixture model are estimated using an EM algorithm.

The third method that we present uses Fisher Linear Discriminant (FLD) to project samples from a high dimensional image space to a lower dimensional feature space. Recently, the Fisherface method has been shown to outperform the widely used Eigenface method in face recognition [10]. The reason for this is that FLD provides a better projection than PCA. In the third proposed method, we decompose the training face and nonface samples into several classes using Kohonen's Self Organizing Map (SOM). From these labeled classes, the within-class and between-class scatter

matrices are computed, thereby generating the optimal projection based on FLD. For each subspace, we use a Gaussian to model each class-conditional density function where the parameters are estimated based on maximum likelihood [41]. To detect faces, each input image is scanned with a rectangular window in which the class-dependent probability is computed. The maximum likelihood decision rule is used to determine whether a face is detected or not.

To capture the variations in face patterns, we use a set of 1,681 face images from Olivetti [171], UMIST [65], Harvard [66], Yale [10] and FERET [141] databases. Both methods have been tested using the databases in [161] [188] to compare their performances with other methods. Our experimental results on the data sets used in [161] [188] (which consist of 225 images with 619 faces) show that our methods detect faces with same reliability as the reported methods in the literature, yet with fewer false detects. To further test our methods, we collect a set of 80 images containing 252 faces. This data set is rather challenging since it contains profile faces, faces with expressions and faces with heavy shadows. Our methods are able to detect most of these faces regardless of their poses, facial expressions and lighting conditions. Furthermore, our methods have fewer false detects than other methods.

1.3 Learning To Recognize 3D Objects

Learning has received increasing attention in recent years. Statistical learning theory has had an influence on many applications ranging from classification and object recognition, grouping and segmentation, illumination modeling, scene reconstruction and others. The rising role of learning methods, made possible by significant improvements in computing power and storage, is largely motivated by the realization that explicit modeling of complex phenomena in a messy world cannot be done without a significant role of learning, both for model and knowledge acquisition, and to support generalization and avoid brittleness. Nevertheless, many statistical and probabilistic learning models require making explicit assumptions, e.g., on the distribution that governs the occurrences of instances in the world. For many visual inference problems such as recognition, categorization and detection, making these assumptions seems unrealistic.

This work develops a distribution free learning theory account to an archetypical visual recognition problem: object recognition. The problem is viewed as that of learning a representation of

an object that, given a new image, is used to recognize the target object in it. The learning account is developed within the PAC (Probably Approximately Correct) model of learnability [202]. This framework allows us to (1) quantify success relative to the distribution of the observed objects, without making assumptions on the distribution. (That is, learnability guarantees that objects sampled from the same distribution as the one that governed the experience of the learner will be recognized correctly.) (2) study the theoretical limits of what can be learned from images in terms of the expressivity of the intermediate representation used by the learning process and (3) develop practical algorithmic solutions to the problem and exhibit their superiority over other methods.

Earlier works have discussed the possibility of identifying the theoretical limits of what can be learned from images [176] and found that learning in terms of the raw representation of the images is computationally intractable. Attempts to explain this focused the dependence of learnability on the representation of the object [43] but failed to provide a satisfactory explanation for it, or a practical solution. The approach developed here builds on suggestions made in [105] and relies heavily on the development of a feature efficient learning approach [115] [22]. That is, a learning process capable of learning in domains in which there is a very large number of potential features but any concept of interest actually depends on a fairly small number of those. At the heart of the learnability approach are two assumptions that we abstract as follows:

Representation: There exists a (possibly infinite) collection $\mathcal{M}$ of "explanations" such that an object can be represented as a simple function of polynomially many elements in $\mathcal{M}$.

Procedural: There exists a process that, given an image that is a positive example of the target object O generates, in polynomial time, "explanations" in $\mathcal{M}$ that are present in the image and such that, with high probability, at least one of them is in the representation of O.

Under these assumptions we prove that there exists an efficient algorithm that can learn a good representation of the object in the sense that, with high probability, it would make correct predictions on future images that contain (or do not contain) the object. Furthermore, we show that under these conditions, the learned representations are robust under realistic noise conditions. A significant non-assumption of our approach is that it has no prior knowledge on the distribution of images nor it is trying to estimate it. Chapter 6 describes this framework in details.

The framework developed here is very general. The *explanations* alluded to above can represent a variety of computational processes and information sources that operate on the image. They can depend on local properties of the image, the relative positions of primitives in the image, and even external information sources or context variables. Thus, the theoretical support given here applies also to an intermediate learning stage in a hierarchical process. The main assumptions of the framework are discussed in Chapter 6.3.

For this framework to contribute to a practical solution, and given our assumptions, there needs to be a computational approach that is able to learn in the presence of a large number of potential "explanations". The evaluation we provide for this framework relies on the SNoW learning architecture [22] that is used here in a large scale object recognition experiment. The SNoW system and the experimental study are described in Chapter 6.

1.4 Geometric Approach to Train Support Vectors Machines

Support Vector Machine (SVM) is a novel machine learning algorithm based on statistical learning theory that can be applied to pattern recognition and regression problems [203] [32]. One distinct characteristic of SVMs is that it aims to find the optimal hyperplane from a set training samples such that the expected recognition error for the unseen test samples is minimized. According to the structural risk minimization inductive principle, a function that describes the training data well and belongs to a set of functions with lowest VC (Vapnik-Chervonenkis) dimension will generalize well regardless of the dimensionality of the input space [32]. Based on this principle, SVM adopts a systematic approach to find a linear function that belongs to a set of functions with lowest VC dimension. The SVM algorithm also provides non-linear function approximations by projecting the input vectors to a high dimensional feature space in which a linear hyperplane is constructed to separate all the projected vectors. One novelty of the nonlinear SVMs is the use of kernel functions to avoid the expensive computations imposed by the nonlinear projection. Although there is no guarantee that the patterns in a high dimensional space can always be linearly separated by a hyperplane, in practice it is usually feasible to find one such linear hyperplane in the projected space.

SVMs have recently attracted much attention because of the rigorous theoretical derivations

from statistical learning theory and excellent empirical results in many classification tasks. Training an SVM is equivalent to solving a quadratic optimization problem in which the support vectors (SVs) are identified to construct the optimal hyperplane. However, many researchers have found that SVMs are not only computationally expensive but also memory intensive.

Consider a training set of m examples, $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, $\ldots$, $(\mathbf{x}_m, y_m)$, where $\mathbf{x}_i$ is an input vector and y_i is its class label, one can solve the following quadratic programming (QP) problem to find the optimal hyperplane:

$$\begin{array}{ll} \text{Minimize} & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i Q_{ij}\alpha_j - \sum_{i=1}^{m}\alpha_i \\ \text{Subject to} & 0 \le \alpha_i \le C \ \text{ and } \ \sum_{i=1}^{m} y_i\alpha_i = 0 \end{array}$$

where C is a parameter for soft margin classifier and Q is an $m \times m$ matrix that depends on the training inputs $\mathbf{x}_i$, the label y_i, and the kernel function of an SVM. Note that a training set of 50,000 examples will yield a Q matrix with 2.5 billion elements, which cannot easily fit into the memory of a standard computer. Solving a large QP problem is usually computationally and memory intensive, nontrivial to implement, and can suffer from numerical instability.

Although several methods have been developed to efficiently solve the quadratic optimization problem [136] [144] [90], none use the *structure* information of the training vectors. Since the support vectors (SVs) form a subset of training vectors which have equal minimum perpendicular distance to the optimal hyperplane and the optimal hyperplane is a weighted linear combination of these SVs, we can construct *exactly* the same optimal hyperplane if we are given only the SVs.

In chapter 7, we propose a method to extract a superset of the SVs based on the structural information of the training vectors. Note that for each support vector $\mathbf{x}_s$ with label y_s, we can always find a hyperplane that passes through $\mathbf{x}_s$ and separates the vectors with label y_s and those with the opposite class label. In other words, a training vector may be a support vector if there exists such a hyperplane that linearly separates all the vectors according to their labels.

The duality theory in computational geometry specifies that a point or a vector in any dimensional space has a unique corresponding hyperplane in the dual space [148], and vice versa. We are interested in determining whether there exits a hyperplane through a vector $\mathbf{x}_i$ in the primal space such that all the points are linearly separated according to their labels in the primal space. Since all

the points in the primal space have dual hyperplanes in the dual space and their normal directions are determined by their labels, the existence of a separating hyperplane at $\mathbf{x}_i$ in the primal space is equivalent to feasibility of a linear program in the dual space. If there exists such a hyperplane, there exists a corresponding point in the dual space. Since the hyperplane through $\mathbf{x}_i$ can linearly separate all the points, $\mathbf{x}_i$ may be a support vector. We will call such vectors the *guard vectors* in the rest of Chapter 7. For an optimal hyperplane with unit normal $\mathbf{w}$, the support vectors must lie on hyperplanes with normalized distance $\frac{1}{||\mathbf{w}||}$ to the optimal hyperplane. Therefore, every support vector must be a guard vector. However, a guard vector may not be a support vector since it may not have minimum distance to the optimal hyperplane.

For a set of m training examples, the superset of SVs are found by solving a set of m linear programming problems. Each LP aims at determining whether a point can be a support vector on the optimal hyperplane. Note that we are interested in the feasibility of each LP rather than the optimal objective value of that LP. After we extract the set of guard vectors, we can construct the same optimal hyperplane by solving a quadratic programming problem with these vectors. Since the guard vectors usually form a small superset of support vectors and linear programming problems can be solved more efficiently than quadratic programming problems, the proposed method is an efficient algorithm to train SVMs. Our experimental results on several benchmarks show that the proposed method has low time and space complexity.

1.5 Thesis Overview

We review the current work in gesture recognition and face detection in Chapter 2. Since there is no survey on face detection in the literature, we give a comprehensive review of the work in this area.

We present our algorithm for extracting and recognizing motion patterns in Chapter 3. Experimental results on extraction and recognition of motion patterns associated with hand gestures using a set of 40 ASL signs are presented. In Chapter 4, we describe a skin color model using a mixture of Gaussians. This model facilitates extraction of skin-tone regions, thereby reducing the computation in gesture recognition and face detection. Three algorithms are introduced in Chapter 5 to detect faces in color and gray-level images. Experimental results on several benchmark databases

are presented. In Chapter 6, we present an account of learnability for object recognition with the PAC framework, a generic learning algorithm, and experimental results based on the theory and the proposed algorithm. In Chapter 7, we propose a geometric approach to train SVMs and compare its performance against conventional methods. We conclude in Chapter 8 with comments on contributions of this thesis and future research.

Chapter 2

Literature Review on Face Detection Methods

With the ubiquity of new information technology and media, more effective and friendly methods for human computer interaction (HCI) are being developed which do not rely on traditional devices such keyboards, mice and displays. Furthermore, the ever decreasing price/performance ratio of computing coupled with recent decreases in video image acquisition cost imply that computer vision systems can be deployed in desktop and embedded systems. The rapidly expanding research in face processing is based on the premise that information about a user's identity, state and intent can be extracted from images and that computers can then react accordingly, e.g., by observing a person's facial expression. In the last five years, face and facial expression recognition have attracted much attention though they have been studied for more than twenty years by psychophysicists, neuroscientists and engineers. Many interesting and useful research demonstrations and commercial applications have been developed from these efforts. A first step of any face processing system is detecting the locations in images where faces are present. However, face detection from a single image or an image sequence is a challenging task because of variability in scale, location, orientation (up-right, rotated), and pose (frontal, profile). Facial expressions, occlusion, and lighting conditions also change the overall appearances of faces.

We now give a definition of *face detection*: Given an arbitrary image or an image sequence, the goal of face detection is to determine whether or not there are any faces in the image, and if present, return their image locations and extents. The challenges associated with face detection can be attributed to the following factors:

- Pose: The images of a face vary due to the relative camera-face pose (frontal, 45 degree, profile, upside down), and some facial features such as an eye or the nose may become partially or wholly occluded.

- Presence or absence of structural components: Facial features such as beards, mustaches and glasses may or may not be present, and there is a great deal of variability amongst these components including shape, color and size.

- Facial expression: The appearance of faces are directly affected by a person's facial expression.

- Occlusion: Faces may be partially occluded by other objects. In an image with a group of people, some faces may partially occlude other faces.

- Image orientation: Face images directly vary for different rotations about the camera's optical axis.

- Imaging conditions : When the image is formed, factors such as lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, lenses) affect the appearance of a face.

There are many closely related problems to face detection. *Face localization* [107] [130] aims to determine the image position of a single face; this is a simplified detection problem with the assumption that the input image contains only one face. The goal of *facial feature detection* [35] is to detect the presence and location of features such as eyes, nose, nostrils, eyebrow, mouth, lips, ears, etc. with the assumption that there is only one face in the image. *Face recognition* or *face identification* [199] [24] [168] compares an input image (probe) against a database (gallery) and reports a match, if any. The purpose of *face authentication* [193] [103] is to verify the claim of the identity of an individual in an input image, while *face tracking* methods continuously estimate the location and possibly the orientation of a face in an image sequence in real time. *Facial expression recognition* concerns identifying the affective states (happy, sad, disgusted, etc) of humans [123] [47] [9] [113]. Evidently, face detection is the first step in any automated system which solves the above problems. It is worth mentioning that many papers use the term "face detection", but the methods and the experimental results indicate that only a single face is localized in an input image.

In this thesis, we differentiate *face detection* from *face localization* since the latter is a simplified problem of the former.

While numerous methods have been proposed to detect faces in a single image or a sequence of intensity or color images, we are unaware of any surveys on this particular topic. A survey of early face recognition methods before 1991 was written by Samal and Iyengar [168]. Chellapa, Wilson, and Sirohey wrote a more recent survey on face recognition and some detection methods [24].

Among the face detection methods, the ones based on learning algorithms have attracted much attention recently and have demonstrated excellent results. Since these data-driven methods rely heavily on the training sets, we will also discuss several databases suitable for this task. A related problem is how to compare the proposed detection methods. Most learning-based detection methods make certain efforts in comparing the performance of several methods, usually in terms of detect and false alarm rates. It is also worth noticing that many metrics have been adopted to evaluate algorithms, such as learning time, execution time, the number of samples required in training, and the ratio between detect rates and false alarms. Evaluation becomes more difficult when researchers use different definitions for detect and false alarm rates. In this chapter, *detect rate* is defined as the ratio between the number of faces correctly detected and the number faces determined by a human. An image region identified as a face by a classifier is correctly detected if the region covers more than α percentage of a face in the image. Most researchers do not explicitly describe such criteria. For example, the reported experimental results in [161] indicate that a face is correctly detected if both eyes and mouth appear in the detected region. A fair evaluation should take these factors into consideration since one can tune the parameters of one's method to increase the detect rates while increasing also the number of false detects. In this chapter, we discuss the benchmarking datasets and the related issues in a fair evaluation.

With over 140 reported approaches to face detection, the research in face detection has broader implications for computer vision research on object recognition. Nearly all model-based or appearance-based approaches to 3-D object recognition have been limited to rigid objects while attempting to robustly perform identification over a broad range of camera locations and illumination conditions. Face detection can be viewed as a two-class recognition problem in which an image region is classified as being a "face" or "non-face". Consequently, face detection is one of the few attempts to

recognize from images (not abstract representations) a class of objects for which there is a great deal of within-class variability (described previously). It is also one of the few classes of objects for which this variability has been captured using large training sets of images, and so some of the detection techniques may be applicable to a much broader class of recognition problems.

Though we report error rates for each method when available, tests are often done on unique datasets, and so comparisons are often difficult. We indicate those methods that have been evaluated with a publically available test set; if we do not indicate the name of the test set, it can be assumed that a unique dataset was used.

We review existing techniques to detect faces from a single intensity or color image. We classify still image detection methods into the following categories; some methods clearly overlap category boundaries.

1. Knowledge-based methods: These rule-based methods hand-coded human knowledge of what constitutes a typical face. Usually, the rules describe the relationships between facial features.

2. Feature invariant approaches: These algorithms aim to find structural features that always exist in the image when pose, viewpoint or lighting conditions do not change drastically.

3. Template matching methods: In template matching, several standard patterns of a face are stored to describe the face as a whole or the facial features separately. The correlations between an input image and the stored patterns are computed for detection.

4. Appearance-based methods: In contrast to template matching, the patterns (or templates) are learned from a large set of training images.

Table 2.1 summarizes algorithms and representative works for face detection in still images within these four categories. Below, we discuss the motivation and general approach of each category. This is followed by a review of specific methods including discussion of their pros and cons. We suggest ways to further improve these methods in Section 2.6.

Table 2.1: A categorization of methods for face detection in still images

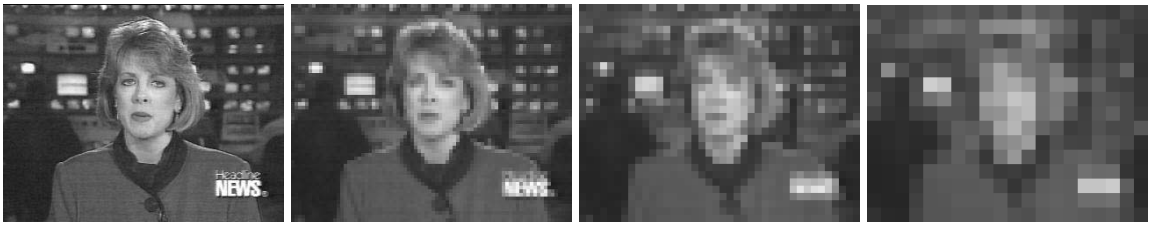
Approach	Representative Works
Knowledge-based	
	Multilevel rule-based method with mosaicing [213]
Feature invariant	
– Facial features	Grouping of edges [110] [228]
– Texture	Space Gray-Level Dependence matrix (SGLD) of face [38]
– Skin Color	Mixture of Gaussian [215] [124]
– Multiple Features	Integration of skin color, size and shape [100]
Template matching	
	Human defined face templates [35] [60]
Appearance-based method	
– Eigenface	Eigenvector decomposition and clustering [199]
– Distribution-based	Gaussian distribution and multilayer perceptron [188]
– Neural Network	Ensemble of neural networks and arbitration [161]
– Support Vector Machine (SVM)	Training SVM with RBF kernel [136]
– Bayesian Approach	Naive Bayes Classifier on local appearance [174]
– Hidden Markov Model (HMM)	Higher order statistics with HMM [153]
– Information-Theoretical Approach	Kullback relative information [111] [30]

2.1 Knowledge-Based Top-Down Methods

In this approach, face detection methods are developed based on the rules derived from the researcher's knowledge of human faces. It is easy to come up with simple rules to describe the features of a face and their relationships. For example, a face often appears in an image with two eyes that are symmetric to each other, a nose and a mouth. The relationships between features can be represented by their relative distances and positions. Facial features in an input image are extracted first. Then face candidates are identified based on the coded rules. A verification process is optionally used to reduce false detections.

One problem with this approach is that it is difficult to translate our knowledge about faces into rules effectively. If the rules are detailed (i.e., strict), they may fail to detect faces that do not pass all the rules. If the rules are too general, they may give many false positives. Moreover, it is difficult to extend this approach to detect faces in different poses since it is impossible to enumerate all the possible cases. On the other hand, heuristics about faces work well in detecting frontal faces in uncluttered scenes.

Yang and Huang [213] use a hierarchical knowledge-based method to detect faces. Their system consists of three levels of rules. At the highest level, all possible face candidates are found by



(a) n=1, original image (b) n=4 (c) n=8 (d) n=16

Figure 2.1: Original and mosaic images at different resolutions. Each square cell consists of $n \times n$ pixels in which the intensity of each pixel is replaced by the average intensity of the pixels in that cell. As n increases, the resolution of cell is lower and contains less detail. Therefore, an algorithm needs to search fewer pixels in a low resolution mosaic image to locate the face.

scanning a window over the input image and applying a set of rules at each location. The rules at a higher level are general descriptions of what a face looks like while the rules at lower levels rely on details of facial features. The rules for levels 1 and 2 are established using mosaic images that are constructed from an original image by decreasing the resolution. An image is divided into square cells, and each cell consists of $n \times n$ pixels. The value of each cell is equal to the average value of the intensity of all the $n \times n$ pixels in the cell. Figure 2.1 shows mosaic images at different resolutions. Examples of coded rules at level 1 such as "the center part of the face (the dark shaded parts in Figure 2.2) has four cells with a basically uniform intensity", "the upper round part of a face (the light shaded parts in Figure 2.2) has a basically uniform intensity", and " the difference between the average gray values of the center part and the upper round part is significant" are used to locate face candidates. Level 1 (highest level) uses a coarse mosaic to search for face candidates, and these are further processed in level 2 using finer mosaics. At level 3, local histogram equalization is performed on the face candidates received from level 2, followed by edge detection. Surviving candidate regions are then examined at level 3 with another set of rules that respond to facial features such as the eyes and mouth. Their test set consists of 60 images. After level 3 the system locates all of the faces in 50 test images (out of the 60 test images) while there are 28 images in which false alarms appear. One attractive feature of this method is the use of mosaic images to detect face candidates at coarser resolutions and then to verify only surviving candidates at finer resolutions. This coarse-to-fine or focus-of-attention strategy reduces the required computation.

Although this method does not result in a high detection rate, the ideas of mosaicing (i.e., multi-resolution), multiple levels, and rules to guide search have been used in later face detection works [102].

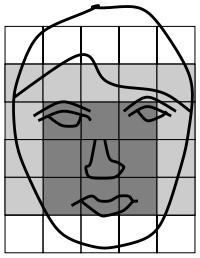
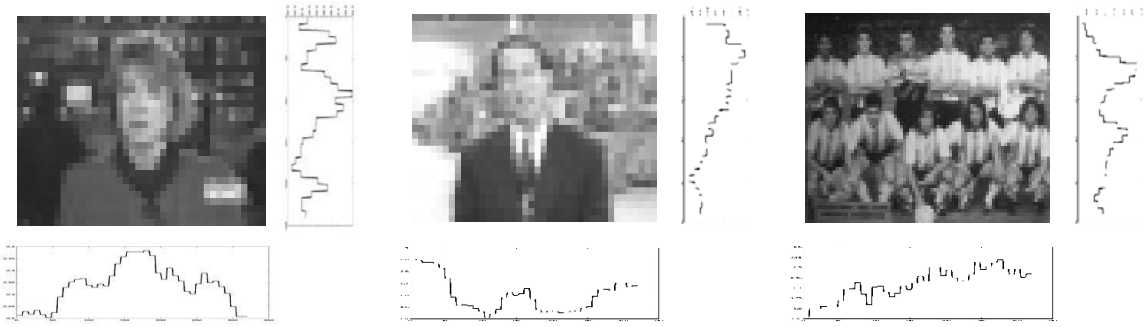


Figure 2.2: A typical face used in knowledge-based top-down methods: Rules are coded based on human knowledge about the characteristics (e.g., intensity distribution and difference) of the facial regions.

Kotropoulos and Pitas [102] present a rule based localization method based on [87] and [213]. First, facial features are located with a projection method that Kanade successfully used [87] to locate the boundary of a face. Let $I(x, y)$ be the intensity value of an $m \times n$ image at position (x, y), the vertical and horizontal projections of the image are defined as $V(I) = \sum_{y=1}^{n} I(x, y)$ and $H(I) = \sum_{x=1}^{m} I(x, y)$. The horizontal profile of an input image is obtained first, and then the two local minima, determined by detecting abrupt changes in $H(I)$, are said to correspond to the left and right side of the head. Similarly, the vertical profile is obtained, and the local minima are determined for the locations of mouth lips, nose tip and eyes. These detected features constitute a facial candidate. Figure 2.3(a) shows one example where the boundaries of the face correspond to the local minimum where abrupt intensity changes occur. Subsequently, eyebrow/eyes, nostrils/nose and the mouth detection rules are used to validate these candidates. The proposed method has been tested using a set of faces with frontal views extracted from the European ACTS M2VTS (MultiModal Verification for Teleservices and Security applications) database [143] which contains video sequences of 37 different people. Each image sequence contains only one face in a uniform background. Their method provides correct face candidates in all tests. The detection rate is 86.5% if successful detection is defined as correctly identifying all facial features. Figure 2.3(b) shows one example in which it becomes difficult to locate a face in complex background using the horizontal and vertical profiles. Furthermore, this method cannot readily detect multiple faces as illustrated in Figure 2.3(c). The projection method can be effective in determining the positions of features if

the window on which they operate is suitably located to avoid misleading interference.



(a) n=8 (b) n=8 (c) n=4

Figure 2.3: Horizontal and vertical profiles of mosaic images. It is feasible to detect a single face by searching for the peaks in horizontal and vertical profiles. However, the same method has difficulty to detect faces in complex backgrounds or multiple faces as shown in 2.3(b) and 2.3(c).

2.2 Bottom-Up Feature-Based Methods

In contrast to the knowledge-based top-down approach, researchers have been trying to find invariant features of faces for detection. The underlying assumption is based on the observation that humans effortlessly detect faces and objects in different poses and lighting conditions, and so there must exist properties or features which are invariant over this variability. Numerous methods have been proposed to first detect facial features and then to infer the presence of a face. Facial features such as eyebrows, eyes, nose, mouth, and hair-line are commonly extracted using edge detectors. Based on the extracted features, some procedure is usually performed to identify their relationships and to verify the existence of a face. The major problem with these feature-based algorithms is that the image features can be severely corrupted due to illumination, noise and occlusion. Feature boundaries can be weak for faces while shadows can cause numerous strong edges which together render perceptual grouping algorithms useless. We describe the feature-based methods in the following categories:

2.2.1 Facial Features

Sirohey [178] proposes a localization method to segment a face from a cluttered background for face identification. It uses the information in the edge map (Canny detector [21]) and heuristics to remove and group edges such that only the ones on the face contour are preserved. An ellipse is then fit to the boundary between the head region and the background. This algorithm achieves 80% accuracy on a database of 48 images with cluttered backgrounds. Chetverikov and Lerch [26] present a simple face detection method using blobs and streaks (linear sequences of similarly oriented edges). Their face model consists of two dark blobs and three light blobs to represent eyes, cheekbones and nose, and streaks to represent the outlines of the faces, eyebrows and lips. Two triangular configurations are utilized to encode the spatial relationship among the blobs. A low resolution Laplacian image is generated to facilitate blob detection. Next, the image is scanned to find specific triangular occurrences as candidates. A face is detected if streaks are identified around a candidate.

Sinha [177] uses a small set of spatial image invariants to describe the space of face patterns. Sinha's key insight for designing the invariant is that while variations in illumination change the individual brightnesses of different parts of faces (such as eyes, cheeks, and forehead), the relative brightnesses of these parts remain largely unchanged. Determining pair-wise ratios of the brightnesses of a few such regions and retaining just the "directions" of these ratios (i.e., is one region brighter or darker than the other?) provides a robust invariant. Thus, observed brightness regularities are encoded as a ratio template which is a coarse spatial template of a face with a few appropriately chosen subregions that roughly correspond to key facial features such as the eyes, cheeks and forehead. The brightness constraints between facial parts are captured by an appropriate set of pairwise brighter-darker relationships between subregions. A face is detected if an image satisfies all the pairwise brighter-darker constraints.

Leung, Burl, and Perona develop a probabilistic method to locate a face in a cluttered scene based on local feature detectors and random graph matching [110]. Their motivation is to formulate the face localization problem as a search problem in which the goal is to find the arrangement of certain facial features that is most likely to be a face pattern. Given a test image, facial features are first extracted using a set of multi-orientation, multiscale Gaussian derivative filters. Candidates

of facial features are identified by matching the filter response at each pixel, $R(x, y)$, against a template (or prototype) vector of responses P_i for i-th facial feature based on the intensity values. Five features (two eyes, two nostrils and nose/lip junction) are used to describe a typical face. Constellations are then formed from the pool of candidates, and the most face-like constellation is determined. Finding the best constellation is formulated as a random graph matching problem in which the nodes of the graph correspond to features on a face, and the arcs represent the distances between different features. Computation of the matching process is reduced by using a controlled search. The basic idea is that given the positions of several features, we can estimate the locations of other features and the covariance of the estimates. Ranking of constellations is based on a probability density function that a constellation corresponds to a face versus the probability it was generated by an alternative mechanism (i.e., nonface). They use a set of 150 images for experiments in which a face is considered correctly detected if any constellation correctly locates three or more features on the faces. This system is able to achieve a correct localization rate of 86%.

Instead of using mutual distances to describe the relationships between facial features in constellations, an alternative method for modeling faces is also proposed by the authors [19]. The representation and ranking of the constellations is accomplished using the statistical theory of shape, developed by Kendall [94] [95], Bookstein [14] [15] and others [121] [40]. The shape statistics is a joint probability density function over N feature points, represented by (x_i, y_i), for i-th feature under the assumption that the original feature points are positioned in the plane according to a general $2N$-dimensional Gaussian distribution. They apply the same maximum likelihood (ML) method to determine the location of a face. One advantage of these methods is that partially occluded faces can be located. However, it is unclear whether they can be adapted to detect multiple faces in a scene.

In contrast to [110] which uses only five feature points to detect faces, Yow and Cipolla [227] [228] propose a feature-based method which uses a large amount of evidence from the visual image and their contextual evidence. The first stage applies a second derivative Gaussian filter, elongated at an aspect ratio of three to one, to a raw image. Interest points, detected at the local maxima in the filter response, indicate the possible locations of facial features. The second stage examines the edges around these interest points and groups them into regions. The perceptual grouping of edges

is based on their proximity and similarity in orientation and strength. Measurements of a region's characteristics, such as edge length, edge strength and intensity variance, are computed and stored into a feature vector. From the training data of facial features, the mean and covariance matrix of each facial feature vector are computed.

An image region becomes a valid facial feature candidate if the Mahalanobis distance between the corresponding feature vectors is below a threshold. The labeled features are further grouped based on model knowledge of where they should occur with respect to each other. Each facial feature and grouping is then evaluated using a Bayesian network. One attractive feature of this method is that it can detect faces at different orientations and poses. The overall detect rate on a test set of 110 images of faces with different scales, orientations and viewpoints is 85% [229]. However the reported false detect rate is 28%, and the implementation is only effective for faces larger than 60×60 pixels. Subsequently, this approach has been enhanced with active contour models [27] [229].

Takacs and Wechsler describe a biologically motivated face localization method which is based on retinal feature extraction and small oscillatory eye movements [192]. The algorithm operates on the conspicuity map, or region of interest, with a retina lattice modeled after the magnocellular ganglion cells in the human vision system. The first phase computes a coarse scan of the image to estimate the location of the face, based on the filter responses of receptive fields. Each receptive field consists of a number of neurons which are implemented with Gaussian filters tuned to specific orientations. The second phase refines the conspicuity map by scanning the image area at a finer resolution to localize the face. The error rate on a test set of 426 images (200 subjects from the FERET database) is 4.69%.

Han et al. develop a morphology-based technique to extract what they call eye-analogue segments for face detection [67]. They argue that eyes and eyebrows are the most salient and stable features of human face and thus useful for detection. They define eye-analogue segments as edges on the contours of eyes. First, morphological operations such as closing, clipped difference, and thresholding are applied to extract pixels at which the intensity values change significantly. These pixels become the eye-analogue pixels in their approach. Then, a labeling process is performed to generate the eye-analogue segments. These segments are used to guide the search for potential face

regions with a geometrical combination of eyes, nose, eyebrows and mouth. The candidate face regions are further verified by a neural network similar to [160]. Their experiments demonstrate a 94% accuracy rate using a test set of 122 images with 130 faces.

Recently Amit, Geman, and Jedynak [4] presented a method for shape detection and apply it to detect frontal-view faces in still intensity images. Detection follows two stages: focusing and intensive classification. Focusing is based on spatial arrangements of edge fragments extracted from a simple edge detector using intensity difference. A rich family of such spatial arrangements, invariant over a range of photometric and geometric transformations, are defined. From a set of 300 training face images, particular spatial arrangements of edges which are more common in faces than backgrounds are selected using an inductive method developed by Amit, Geman, and Wilder [5]. Meanwhile, the standard CART algorithm [17] is applied to grow a classification tree from the training images and a collection of false positives identified on generic background images. Given a test image, the regions of interest are identified based on the spatial arrangements of edge fragments. Each region of interest is then classified as face or background using the learned CART tree [17]. Their experimental results on a set of 100 images from Olivetti (now AT & T) data set [171] report a false positive rate of 0.2% per 1000 pixels and a false negative rate of 10%.

2.2.2 Texture

Human faces have a distinct texture that can be used to separate them from different objects. Augusteijn and Skufca [8] develop a method to infer the presence of a face through the identification of face-like textures. The texture are computed using the second-order statistical features [68] on subimages of 16×16 pixels. Three types of features are considered: skin, hair and others. They use a cascade-correlation neural network [48] for supervised classification of textures and a Kohonen self-organizing feature map [101] to show the clustering of the difference texture classes. To infer the presence of a face from the texture labels, they suggest using votes of the occurrence of hair and skin textures. However only the result of texture classification are reported, not face localization or detection.

Dai and Nakano [38] develop a method based on the feature parameters of space gray-level dependence matrix (SGLD) [68]. A SGLD matrix counts the concurrence of displaced pixels which

have a certain pair of intensity values. Based on SGLD matrix, a set of textural features is derived to describe the characteristics of a face. A face is defined as a region where certain inequalities of the feature vectors hold. Color information is also incorporated with the face-texture model. Using the face texture model, they design a scanning scheme for face detection in color scenes, in which the orange-like parts including the face areas are enhanced by utilizing the I component of the YIQ color space. One advantage of this approach is that it can detect faces which are not upright or have features such as beards and glasses. The reported detection rate is perfect for 60 faces in 30 test images.

2.2.3 Skin Color

Human skin color has been used and proven to be an effective feature in many applications from face detection to hand tracking. Although different people have different skin color, several studies have shown that the major difference between lies in their intensity rather than their chrominance [64] [215]. Several color spaces have been utilized to label pixels as skin including RGB [80] [81] [172], normalized RGB [36] [210] [183] [215] [37] [134] [214] [97] [185] [151], HSV (or HSI) [173] [100] [181] [180], YCrCb [207] [23], YIQ [38], YES [166], CIE XYZ [25], and CIE LUV [216]. Although it is still arguable which color space is most suitable for face detection, many researchers agree that a skin detection is more effective when the intensity component is discarded.

Many methods have been proposed to build a skin color model. The simplest model is to define a region of skin tone pixels using Cr, Cb values [23], i.e., $R(Cr, Cb)$, from samples of skin color pixels. With carefully chosen thresholds, $[Cr_1, Cr_2]$ and $[Cb_1, Cb_2]$, a pixel is classified to have skin tone if its values (Cr, Cb), fall within the ranges, i.e., $Cr_1 \leq Cr \leq Cr_2$ and $Cb_1 \leq Cb \leq Cb_2$. Crowley and Coutaz [36] [37] use a histogram $h(r, g)$ of (r, g) values in normalized RGB color space to obtain the probability of obtaining a particular RGB-vector given that the pixel observes skin. In other words, a pixel is classified to belong to skin color if $h(r, g) \geq \tau$ where τ is a threshold selected empirically from the histogram of samples. Saxe and Foulds [173] propose an iterative skin identification method that uses histogram intersection [189] in HSV color space. An initial path of skin color pixels, called the control seed, is chosen by the user and is used to initiate the iterative algorithm. To detect skin color regions, their method moves through the image, one

patch at a time, and presents the control histogram and the current histogram from the image for comparison. Histogram intersection, described in detail by Swain and Ballad [189] is used to compare the control histogram and current histogram. If the match score or number of instances in common (i.e., intersection) is greater than a preselected threshold, then the current patch of pixels is classified as being skin color. Kjeldsen and Kender [100] define a color predicate in HSV color space to separate skin regions from background. In contrast to the nonparametric methods mentioned above, Gaussian density functions [20] [97] [216] and a mixture of Gaussians [80] [81] [219] are often used to model skin color. The parameters in a unimodal Gaussian distribution are often estimated using maximum likelihood [20] [97] [216]. The motivation for using a mixture of Gaussians is based on the observation that the color histogram of colors for the skin of people with different ethnic background does not form a unimodal distribution, but rather a multimodal distribution. The parameters in a mixture of Gaussians are usually estimated using an EM algorithm [80] [219]. Recently, Jones and Rehg [85] conduct a large-scale experiment in which nearly 1 billion labeled skin tone pixels are collected (in normalized RGB color space). After comparing the performance of histogram and mixture models for skin detection, they find histogram models to be superior in accuracy and computational cost.

Color information is an efficient tool for identifying facial areas and specific facial features if the skin color model can be properly adapted for different lighting environments. However, such skin color models are not effective where the spectrum of the light source varies significantly. In other words, color appearance is often unstable due to changes in both background and foreground lighting. Though the color constancy problem has been addressed through the formulation of physics-based models of light formation [52], several approaches have been proposed to use skin color in varying lighting conditions. McKenna, Raja, and Gong [125] present an adaptive color mixture model to track faces under varying illumination conditions. Instead of relying on a skin color model based on color constancy, they use a stochastic model to estimate an object's color distribution on-line and adapt to accommodate changes in the viewing and lighting conditions. Preliminary results show that their system can track faces within a range of illumination conditions. However, this method cannot be applied to detect faces in still images.

Skin color alone is usually not sufficient to detect or track faces. Recently several modular

systems using a combination of shape analysis, color segmentation and motion information for locating or tracking heads and faces in an image sequence have been developed [64] [216] [215] [125] [181]. See Section 2.2.4 for more detail.

2.2.4 Multiple Features

Recently, numerous methods that combine several facial features have been proposed to locate and detect faces. Most of them utilize global features such as skin color, size, and shape to find face candidates and then verify these candidates using local, detailed features such as eye brows, nose and hair. A typical approach begins with detection of skin-like regions as described in Section 2.2.3. Next, skin-like pixels are grouped together using connected component analysis or clustering algorithms. If the shape of a connected region has an elliptic or oval shape, it becomes a face candidate. Finally, local features are used for verification.

Yachida et al. present a method to detect faces in color images using fuzzy theory [25] [212] [211]. They use two fuzzy models to describe the distribution of skin and hair color in CIE XYZ color space. Five (one frontal, and four side views) head-shape models are used to abstract the appearance of faces in images. Each shape model is a two dimensional pattern consisting of $m \times n$ square cells. They assign two properties to each cell: the skin proportion and the hair proportion, which indicate the ratios of the skin area (or the hair area) within the cell to the area of the cell (since each face image is divided into $m \times n$ cells, each cell may contain several pixels). In a test image each pixel is classified as hair, face, hair/face, hair/background based on the distribution models, thereby generating skin-like and hair-like regions. The head-shape models are then compared with the extracted skin-like and hair-like regions in a test image. If they are similar, the detected region becomes a face candidate. For verification, eye-eyebrow and nose-mouth features are extracted from a face candidate using horizontal edges.

Sobottka and Pitas [181] propose a method for face localization and facial feature extraction using shape and color. First, color segmentation in HSV space is performed to locate skin-like regions. Connected components are then determined by region growing at a coarse resolution of the segmented image. For each connected component, the best-fit ellipse is computed using geometric moments. Connected components that are well approximated by an ellipse are selected

as face candidates. Subsequently these candidates are verified by searching for facial features inside of the connected components. Features, such as eyes and mouths, are extracted based on the observation that they are darker than the rest of a face.

Saber and Tekalp [166] present a frontal-view face localization method based on color, shape and symmetry. Skin/non-skin classification is carried out using the class-conditional density function in YES color space followed by Gibbs random field model-based smoothing in order to yield contiguous regions. Next, an elliptical face template is used to determine the similarity of the skin color regions based on Hausdorff distance [78]. Finally, the eye centers are localized using several cost functions which are designed to take advantage of the inherent symmetries associated with face and eye locations. The tip of the nose and the center of the mouth are then located by utilizing the distance between the eye centers. One drawback is that it is effective only for a single frontal-view face and when both eyes are visible. A similar method that uses color and local symmetry is presented in [185].

Yang and Ahuja [216] propose a detection method that is based on structure, color and geometry. First, multiscale segmentation [2] is performed to extract homogeneous regions in an image. Using a Gaussian skin color model, regions of skin tone are extracted and grouped into ellipses. A face is detected if facial features such as eyes and mouth exist in the elliptic regions of skin tone. Experimental results show that this method is able to detect faces at different orientations with facial features such as beard and glasses.

Terrillon, David, and Akamastu [194] [195] develop a method based on color and shape analysis. Using a Gaussian skin color model, a binary image of skin/non-skin tone pixels is generated. To characterize the shape of the clusters in the binary image, a set of 11 lowest-order geometric moments is computed using Fourier and radial Mellin transforms. For detection, a neural network is trained with the extracted geometric moments. Their experiments show a detect rate of 85% based on a test set of 100 images.

Pentland and Kauth propose a blob representation to extract a compact, structurally meaningful description of multispectral satellite imagery [91]. A feature vector at each pixel is formed by concatenating the pixel's image coordinates to the pixel's spectral (or textural) components; pixels are then clustered using this feature vector to form coherent connected regions, or "blobs." To

detect faces, each feature vector consists of the image coordinates and normalized chrominance, i.e., $X = (x, y, \frac{r}{r+g+b}, \frac{g}{r+g+b})$ [183] [134]. A connectivity algorithm is then used to grow blobs, and the resulting skin blob whose size and shape is closest to that of a canonical face is considered as a face.

To locate a face in a color image that contains a person's head and shoulders, Chai and Ngan [23] present a method based on color and intensity characteristics. First, color segmentation is performed to extract pixels of skin tones using a simple reference map in YCrCb color space. The color segmentation algorithm is based on a pre-defined region of Cr and Cb. Morphological operations such as dilation and erosion are used to remove noise, followed by a simple clustering method to group pixels of skin color. For their applications, they assume the intensity in the facial region is non-uniform while the intensity in the background tends to have a more even distribution. Based on this assumption, they classify the skin-tone cluster with non-uniform intensity as a face region. This is followed by some correction geometric methods to locate a single face. One drawback of this approach is that it can only be applied in limited domains because of its assumptions on intensity distributions of facial regions and background.

Hotta, Kurita, and Mishima present a detection method that uses higher-order local autocorrelation (HLAC) features [76]. First, each input image is transformed to log-polar coordinates. Next, a set of 35 mask patterns of 3×3 pixels is used to compute HLAC features from the log-polar images. HLAC features are extracted from the training face and nonface samples, and linear discriminant analysis [41] is applied to find a projection that maximizes the ratio of between-class scatter to within-class scatter. In the projected space, the distance between the mean feature vector of all face samples and the feature vector of the input sample is computed. A face is detected if the distance is below a threshold. Their experiments show high detect rate on the training set. However, the method has not been tested for other unseen images.

Range and color have also been employed for face detection by Kim et al. [97]. Disparity maps are computed and objects are segmented from the background with a disparity histogram using the assumption that background pixels have the same depth and they outnumber the pixels in the foreground objects. Using a Gaussian distribution in normalized RGB color space, segmented regions with a skin-like color are classified as faces.

2.3 Template Matching

In template matching, a standard pattern of a face (usually frontal) is manually defined. Given an input image, the correlation values with several sizes of the standard pattern are calculated for the face contour, eyes, nose and mouth independently. The existence of a face is determined using the correlation values. This approach has the advantage of being simple to implement. However, it has proven to be inadequate for face detection since it cannot deal with variation in scale, pose and shape. Multiresolution, multiscale, subtemplates and deformable templates have subsequently been proposed to achieve scale and shape invariance.

2.3.1 Simple Templates

An early attempt to detect frontal faces in photographs is reported by Sakai, Nagao, and Fujibayashi [167]. They use several subtemplates for the eyes, nose, mouth and face contour to model a face. Each subtemplate is defined in terms of line segments. Lines in the input image are extracted based on greatest gradient change and then matched against the subtemplates. The correlations between subimages and contour template are computed first to detect the candidate locations of faces. Then matching with the other subtemplates is performed at the candidate positions. In other words, the first phase determines focus of attention or region of interest, and the second phase examines the details to determine the existence of a face. The idea of focus of attention and subtemplates has been adopted by later works on face detection.

Craw, Ellis, and Lishman present a localization method based on a shape template of a frontal-view face [34]. A Sobel filter is first used to extract edges. These edges are grouped together to search for the template of a face based on several constraints. After the head contour has been located, the same process is repeated at different scales to locate features such as eyes, eyebrows and lips. Craw, Tock, and Bennett [35] later describe a localization method using a set of 40 templates to search for facial features and a control strategy to guide and assess the results from the template-based feature detectors.

Tsukamoto, Lee, and Tsuji present a qualitative model for face (QMF) [197] [198]. In QMF, each sample image is divided into N blocks, and qualitative features are estimated for each block. To parameterize a face, "lightness" and "edgeness" are defined as the features in this model.

Consequently, this blocked template is used to calculate "faceness" at every position of an input image. A face is detected if the faceness measure is above a predefined threshold.

Samal and Iyengar [169] use the outlines (silhouettes) to locate faces. A set of basis face silhouettes is obtained using principal component analysis (PCA) on face examples in which the silhouette is represented by an $n \times n$ array of bits. These eigen-silhouettes are then used with a generalized Hough transform for localization.

Sumi and Ohta [184] propose a localization method based on multiple templates for facial components. They define numerous hypotheses for the possible appearances of facial features. A set of hypotheses for the existence of a face is then defined in terms of the hypotheses for facial components using the Dempster-Shafer theory [39]. Given an image, feature detectors compute confidence factors for the existence of facial features. The confidence factors are combined to determine the measures of belief and disbelief about the existence of a face. Their system is able to locate faces in 88 images out of 94 images.

A hierarchical template matching method for face detection is proposed by Miao et al. [128]. At the first stage, an input image is rotated from -20° to 20° in steps of 5° in order to detect rotated faces. These images then form mosaic at different scales in which edges are extracted using the Laplacian operator (See Figure 2.1 for examples of mosaic images.). The face template consists of six facial components of two eyebrows, two eyes, one nose, and one mouth. Face candidates are located by matching templates of face models represented in edges. Finally, heuristics are applied to determine the existence of a face. Their experimental results show better results in images containing a single face (frontal or rotated) than images with multiple faces.

2.3.2 Deformable Templates

Govindaraju et al. [61] [62] [60] present a two stage face detection method in which face hypotheses are generated and tested. A face model is built in terms of features defined by the edges. These features describe the curves of the left side, the hair-line and the right side of a frontal face. The Marr-Hildreth edge operator is used to obtain an edge map of an input image. A filter is then used to remove objects whose contours are unlikely to be parts of a face. Pairs of fragmented contours are linked based on their proximity and their relative orientations. Corners in the contours are

detected to segment them into feature curves. These feature curves are then labeled by checking their geometric properties and relative positions in the neighborhood. Pairs of feature curves are joined by edges if their attributes are compatible (i.e., if they could arise from the same face). The ratios of the feature pairs forming an edge is compared with the golden ratio, and a cost is assigned to the edge. If the cost of a group of three feature curves (with different labels) is low, the group becomes a hypothesis. Collateral information, which indicates the number of persons in the image, is obtained from the caption of the input image to select the best hypotheses. Their system reports a detect rate of approximately 70% based on a test set of 50 photographs. However the faces must be upright, unoccluded, and frontal. The same approach has been extended by extracting edges in the wavelet domain by Venkatraman and Govindaraju [204].

Yuille, Hallinan, and Cohen [230] use deformable templates to model facial features that fit an a priori elastic model to facial features (e.g., eyes). In this approach, facial features are described by parameterized templates. An energy function is defined to link edges, peaks and valleys in the input image to corresponding parameters in the template. The best fit of the elastic model is found by minimizing an energy function of the parameters. Although their experimental results demonstrate good performance in tracking nonrigid features, one drawback of this approach is that the deformable template must be initialized in the proximity of the object of interest.

In [106] Kwon and da Vitoria Lobo propose a detection method based on snakes [89] [112] and templates. An image is first convolved with a blurring filter and then a morphological operator to enhance edges. A modified n-pixel (n is small) snake is used to find and eliminate small curve segments. Each face is approximated by an ellipse, and a Hough transform of the remaining snakelets is used to find a dominate ellipse. Thus, sets of four parameters describing the ellipses are obtained and used as candidates for face locations. For each of these candidates, a method similar to the deformable template method [230] is used to find detailed features. If a substantial number of the facial features are found successfully and their proportion satisfy ratio tests based on a face template, a face is considered to be detected. Lam and Yan [107] also use snakes to locate the head boundaries with a greedy algorithm in minimizing the energy function.

Lanitis, Taylor, and Cootes describe a face representation method with both shape and intensity information [108]. They start with sets of training images in which sampled contours such as the

eye boundary, nose chin/cheek are manually labeled, and the vector sample points is used as shape features to be detected. They use a point distribution model (PDM) to characterize the shape vectors over an ensemble of individuals, and an approach similar to Kirby and Sirovich [98] to represent shape-normalized intensity appearance. A face-shape PDM can be used to locate faces in new images by using active shape model (ASM) search to estimate the face location and shape parameters. The face patch is then deformed to the average shape, and intensity parameters are extracted. The shape and intensity parameters can be used together for classification. Cootes and Taylor apply a similar approach to localize a face in an image [31]. First, they define a rectangular regions of the image containing instances of the feature of interest. Factor analysis [6] is then applied to fit these training features and obtain a distribution function. Candidate features are determined if the probabilistic measures are above a threshold, and are verified using the ASM. After training this method with 40 images, it is able to locate 35 faces in 40 test images. The ASM approach has also been extended with two Kalman filters [44] [45] to estimate the shape-free intensity parameters and used to track faces in image sequences.

2.4 Appearance-Based Methods

Contrasted to the template matching methods where the templates are pre-defined by experts, the "templates" in appearance-based methods are learned from the examples in the images. In general, appearance-based methods rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face and non-face images. The learned characteristics are in the form of distributions, models or discriminant functions that are consequently used for face detection. Meanwhile, dimensionality reduction is usually carried out because of efficiency and efficacy.

Many appearance-based methods can be understood in a probabilistic framework. An image or feature vector derived from an image is viewed as a random variable $\mathbf{x}$, and this random variable is characterized for faces and non-faces by the class-conditional density functions $p(\mathbf{x}|face)$ and $p(\mathbf{x}|nonface)$. Bayesian classification or maximum likelihood can be used to classify a candidate image location as face or non-face. Unfortunately, a straightforward implementation of Bayesian classification is infeasible because of the high dimensionality of $\mathbf{x}$, because $p(\mathbf{x}|face)$ and $p(\mathbf{x}|nonface)$ are multimodal, and because it is not yet understood if there are natural parameter-

ized forms for $p(\mathbf{x}|face)$ and $p(\mathbf{x}|nonface)$. Hence, much of the work in appearance-based method concerns empirically validated parametric and non-parametric approximations to $p(\mathbf{x}|face)$ and $p(\mathbf{x}|nonface)$.

2.4.1 Eigenfaces

Kirby and Sirovich [98] demonstrate that images of faces can be linearly encoded using a modest number of basis images. This demonstration is based on the Karhunen-Loève transform [88] [117] [57], which also goes by other names, e.g., principal component analysis [84] and the Hotelling transform [59]. The idea is arguably proposed first by Pearson in 1901 [138] and then by Hotelling in 1933 [75]. Given a collection of n by m pixel training images represented as a vector of size $m \times n$, basis vectors spanning an optimal subspace are determined such that the mean square error between the projection of the training images onto this subspace and the original images is minimized. They call the set of optimal basis vectors eigenpictures since these are simply the eigenvectors of the covariance matrix computed from the vectorized face images in the training set. Experiments with a set of 100 images show that a face image of 91×50 pixels can be effectively encoded using only 50 eigenpictures, while retaining a reasonable likeness (i.e., capturing 95% of the variance).

Turk and Pentland applied principal component analysis to face recognition and detection [199]. Similar to [98], principal component analysis on a training set of face images is used to generate the Eigenpictures (here called Eigenfaces) which span a subspace (called the face space) of the image space. Images of faces are projected onto the subspace and clustered. Similarly, nonface training images are projected onto the same subspace and clustered. Since images of faces do not change radically when projected onto the face space, while the projection of nonface images appear quite different. To detect the presence of a face in a scene, the distance between an image region and the face space is computed for all locations in the image. The distance from face space is used as a measure of "faceness", and the result of calculating the distance from face space is a "face map". A face can then be detected from the local minima of the face map. Many works on face detection, recognition, and feature extractions have adopted the idea of eigenvector decomposition and clustering.

2.4.2 Distribution-Based Methods

Sung and Poggio develop a clustering and distribution-based system for face detection [186] [187] [188] which extends Sinha's [177] invariance-based system by using a learning approach, rather than manually encoding the invariants. Their system consists of two components, distribution-based models for face/nonface patterns and a multilayer perceptron classifier. Each face and nonface example is first normalized and processed to a 19×19 pixel image and treated as a 361-dimensional vector or pattern. Next, the patterns are grouped into six face and six nonface clusters as shown in Figure 2.4. Each cluster is represented as a multidimensional Gaussian function with a mean image and a covariance matrix. Two distance metrics are computed between an input image pattern and the prototype clusters. The first distance component is the normalized Mahalanobis distance between the test pattern and the cluster centroid, measured within a lower-dimensional subspace spanned by the cluster's 75 largest eigenvectors. The second distance component is the Euclidean distance between the test pattern and its projection onto the 75-dimensional subspace. This distance component accounts for pattern differences not captured by the first distance component. The last step is to use a multilayer perceptron network to classify face window patterns from non-face patterns using the twelve pairs of distances to each face and non-face cluster. The classifier is trained using standard backpropagation from a database of 47,316 window patterns. There are 4,150 positive examples of face patterns and the rest are nonface patterns. Note that it is easy to collect a representative sample face patterns, but much more difficult to get a representative sample of nonface patterns. This problem is avoided by a bootstrap method that electively adds images to the training set as training progress. They start with a small set of nonface examples in the training example and train the MLP classifier with the current database of examples. Then, they run the face detector on a sequence of random images and collect all the nonface patterns that the current system wrongly classifies as faces. These nonface patterns are then added to the training database as new nonface examples. This bootstrap method avoids the problem of collecting a representative sample of nonface patterns.

A probabilistic visual learning method, based on density estimation in a high-dimensional space using an eigenspace decomposition is developed by Moghaddam and Pentland [130]. Principal component analysis is used to define the subspace best representing a set of face patterns. These

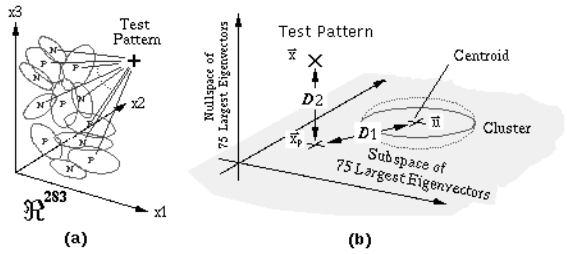


Figure 2.4: Face and nonface clusters used by Sung and Poggio. Their method estimates density functions for face and nonface patterns using a set of Gaussians. The estimated distributions are then used to compute the Mahalanobis distances from each image pattern to the cluster centers. These distances are then used for detection.

principal components preserve the major linear correlations in the data and discard the minor ones. This method decomposes the vector space into two mutually exclusive and complementary subspaces: the principal subspace (or feature space) and its orthogonal complement. Therefore, the target density is decomposed into two components: the density in the principal subspace (spanned by the principal components) and its orthogonal complement (which is discarded in standard principal component analysis). A multivariate Gaussian and a mixture of Gaussians are used to learn the statistics of the local features of a face. These probability densities are then used for object detection based on maximum likelihood estimation. The proposed method has been applied to face localization, coding and recognition. Compared with the classic eigenface approach [199], the proposed method shows better performance in face recognition. In terms of face detection, this technique has only been demonstrated on localization (i.e., input images contain only one face); see also [96].

Yang, Ahuja, and Kriegman propose two distribution-based methods [223]. The first method is an extension of factor analysis [6]. Factor analysis (FA), a statistical method for modeling the covariance structure of high dimensional data using a small number of latent variables, has analogies with principal component analysis. However PCA, unlike FA, does not define a proper density model for the data since the cost of coding a data point is equal anywhere within the principal component subspace (i.e., the density is unnormalized along these directions). Further, PCA is not robust to independent noise in the features of the data since the principal components maximize the variances of the input data, thereby retaining unwanted variations. Hinton, Dayan, and Revow have applied FA to digit recognition, and they compare the performance of PCA and

Figure 2.5: Prototype of each face class using Kohonen's SOM by Yang, Ahuja and Kriegman. Each prototype corresponds to the center of a cluster.

FA models [72]. A mixture model of factor analyzers has recently been extended [58] and applied to face recognition [54]. Both studies show that FA performs better than PCA in digit and face recognition. Since pose, orientation, expression, and lighting conditions introduce variances in the face patterns, the distribution can be better modeled using a mixture of subspaces in which each subspace captures certain variance of patterns. A mixture of factor analyzers (MFA) is used to model the distribution of face patterns in which the parameters are estimated using an EM algorithm [39] [154]. A face is detected if the likelihood of there being a face is above a threshold.

The second method in [223] uses Fisher Linear Discriminant (FLD) to project samples from a high dimensional image space to a lower dimensional feature space. Recently, multidiscriminant analysis (Fisherface) has been shown to outperform the widely used Eigenface method in face recognition [10] [231] [190] [232]. Its success is due to the fact that FLD provides a better projection for pattern classification than PCA in face recognition [10]. In this method, the training face and nonface sets are decomposed into several classes using Kohonen's Self Organizing Map (SOM). Figure 2.5 shows a prototype of each face class. From these labeled classes, the within-class and between-class scatter matrices are computed, to provide the projection matrix based on FLD. For each class, a Gaussian distribution is used to model the class-conditional density function, and the parameters are estimated based on maximum likelihood [41]. To detect faces, each input

image is scanned with a rectangular window and the class-dependent probability is computed. The maximum likelihood decision rule is used to determine whether a face is detected or not. Both methods in [223] have been tested using the databases in [161] [188] which consist of 225 images with 619 faces, and experimental results show that these two methods have comparable detection rates (92.3% for MFA and 93.6% for FLD-based method. to the best results reported in the literature, yet with fewer false detects (See Table 2.4 for comparison).

2.4.3 Neural Networks

Neural networks have been applied successfully in many pattern recognition problems such as optical character recognition [109], object recognition [145] and autonomous robot driving [146]. Since face detection can be treated as a two class pattern recognition problem, various neural network architectures have been proposed. The advantage of using neural networks in face detection is the feasibility of training a system to capture the complex class conditional density of face patterns. However, one drawback is that the network architecture has to be extensively tuned (number of layers, number of nodes, learning rates, etc.) to get exceptional performance.

An early method using hierarchical neural networks is proposed by Agui et al. [1]. The first stage consists of two parallel subnetworks in which the inputs are intensity values from an original image and intensity values from an edge emphasized (using a 3×3 Sobel filter) image. The inputs to the second stage network consist of the outputs from the subnetworks and extracted feature values such as the standard deviation of the pixel values in the input pattern, a ratio of the number of white pixels to the total number of binarized pixels in a window, and geometric moments. An output value at the second stage indicates the presence of a face in the input region. Experimental results show that this method is able to detect faces if all faces in the test images are almost of the same size. Propp and Samal [149] developed one of the earliest neural networks for face detection. Their network consists of 4 layers with 1024 input units, 256 units in the first hidden layer, 8 units in the second hidden layer, and 2 output units. A similar hierarchical neural network is later proposed by [86]. The early method by Soulie, Vinnet, and Lamy [182] scans an input image with a time-delay neural network [206] (with a receptive field of 20×25 pixels) to detect faces. To cope with size variation, the input image is decomposed using wavelet transforms. They report a false

negative rate of 2.7% and false positive rate of 0.5% from a test of 120 images. In [201], Vaillant, Monrocq and Le Cun use convolutional neural networks to detect faces in images. Examples of face and nonface images of 20×20 pixels are first created. One neural network is trained to find rough localizations of faces at some scale. Another network is trained to determine the exact position of faces at some scale. Given an image, areas which may contain faces can be selected as face candidates by the first network. These candidates are verified by the second network. Burel and Carel [18] propose a neural network for face detection in which the large number of training examples of faces and nonfaces are compressed into fewer examples using a Kohonen algorithm, i.e., vector quantization. A multi-layer perceptron is used to learn these examples for face/background classification. The detection phase consists of scanning each image at various resolution. For each location and size of the scanning window, the contents are normalized to a standard size, and the intensity mean and variance are scaled to reduce the effects of lighting conditions. Each normalized window is then classified by a MLP.

Feraud and Bernier [50] [49] present a detection method using autoassociative neural networks. The idea is based on [104] which shows an autoassociative network with five layers is able to perform a nonlinear principal component analysis. One autoassociative network is used to detect frontal-view faces and another one is used to detect turned faces up to 60 degrees to the left and right of the frontal view. A gating network is also utilized to assign weights to frontal and turned face detectors in an ensemble of autoassociative networks. On a small test set of 42 images, they report a detection rate similar to [159]. The method has also been employed in LISTEN [28] and MULTRAK [11].

Lin, Kung, and Lin [114] present a face detection system using probabilistic decision-based neural network (PDBNN). The architecture of PDBNN is similar to radial basis function (RBF) network with modified learning rules and probabilistic interpretation. Instead of converting a whole face image into a training vector of intensity values for the neural network, they first extract feature vectors based on intensity and edge information in the facial region that contains eyebrows, eyes and nose. The extracted two feature vectors are fed into two PDBNN's and the fusion of the outputs determine the classification result. Based on a set of 23 images provided by Sung and Poggio [188], their experimental results show comparable performance with the other leading neural network

based face detectors [188] [161].

Among all the face detection methods that use neural networks, the most significant work is probably by Rowley, Baluja, and Kanade [160] [159] [161]. Their method has some similarities to the system by Sung and Poggio [187] and [18]. A multilayer neural network is used to learn the face and nonface patterns from face/nonface images (i.e., the intensities and spatial relationships of pixels) while Sung and Poggio [187] use a neural network to find a discriminant function to classify face and nonface using distance measures. They also use multiple neural networks and several arbitration methods to improve performance while Burel and Carel [18] used a single network and Vaillant, Monrocq, and Le Cun [201] used two networks for classification. There are two major components: multiple neural networks (to detect face patterns) and a decision-making module (to render the final decision from multiple detection results). As shown in Figure 2.6, the first component of this method is a neural network that receives a 20×20 pixel region of an image and outputs a score ranging from -1 to 1. Given a test pattern, the output of the trained neural network indicates the evidence for a nonface (close to -1) or face pattern (close to 1). To detect faces anywhere in an image, the neural network is applied at all image locations. To detect faces larger than 20×20 pixels, the input image is repeatedly subsampled, and the network is applied at each scale. Nearly 1,050 face samples of various sizes, orientations, positions and intensities are used to train the network. In each training image, the eyes, tip of the nose, corners and center of the mouth are labeled manually and used to normalize the face to the same scale, orientation, and position. The second component of this method is to merge overlapping detection and arbitrate between the outputs of multiple networks. Simple arbitration schemes such as logic operators (AND/OR) and voting are used to improve performance. Rowley, Baluja, and Kanade [160] report several systems with different arbitration schemes that are less computationally expensive than Sung and Poggio's system, and have higher detect rates based on a test set of 24 images containing 144 faces.

One limitation of the methods by Rowley [160] and by Sung [187] is that they can only detect upright, frontal faces. Recently, Rowley [162] extended this method to detect rotated faces using a router network which processes each input window to determine the possible face orientation and then rotates the window to a canonical orientation; the rotated window is presented to the neural networks as described above. However, the new system has a lower detection rate on upright faces

than the upright detector. Nevertheless, the system is able to detect 76.9% of faces over two large test sets with a small number of false positives.

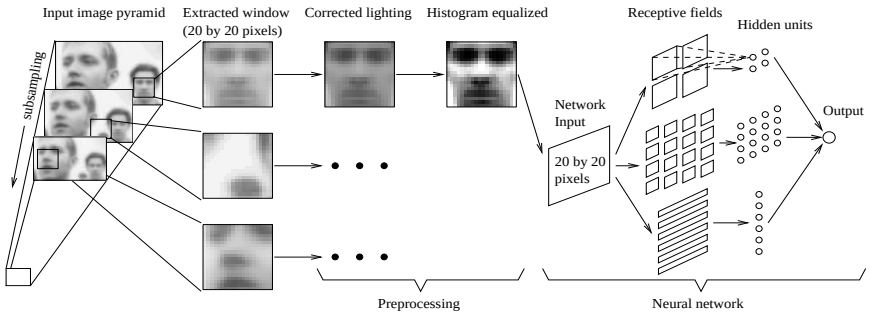


Figure 2.6: System diagram of Rowley's method. Each face is pre-processed before feeding it to an ensemble of neural networks. Several arbitration methods are used to determine whether a face exists based on the output of these networks.

2.4.4 Support Vector Machines

Support Vector Machines (SVMs) were first applied to face detection by Osuna, Freud, and Girosi [136]. SVMs can be considered as a new paradigm to train polynomial function, neural networks, or radial basis function (RBF) classifiers. While most methods for training a classifier (e.g., Bayesian, neural networks, and RBF) are based on of minimizing the training error, i.e., *empirical risk*, SVMs operates on another induction principle, called *structural risk minimization*, which aims to minimize an upper bound on the expected generalization error. An SVM classifier is linear classifier where the separating hyperplane is chosen to minimize the expected classification error of the unseen test patterns. This optimal hyperplane is defined by a weighted combination of a small subset of the training vectors, called support vectors. Estimating the optimal hyperplane is equivalent to solving a linearly constrained quadratic programming problem. However, the computation is both time and space intensive. In [136] Osuna, Freund and Girosi develop an efficient method to train an SVM for large scale problems, and apply it to face detection. Based on two test sets of 10,000,000 test patterns of 19×19 pixels, their system has slightly lower error rates and runs approximately 30 times faster than the system by Sung and Poggio [186]. SVMs have also been used to detect faces and pedestrians in the wavelet domain [135] [137].

2.4.5 Sparse Network of Winnows

Yang, Roth and Ahuja propose a method that uses SNoW learning architecture [157] [22] to detect faces with different features and expressions, in different poses, and under different lighting conditions. They also study the effect of learning with primitive as well as with multi-scale features. SNoW (Sparse Network of Winnows) is a sparse network of linear functions that utilizes the Winnow update rule [115]. SNoW is specifically tailored for learning in domains in which the potential number of features taking part in decisions is very large, but may be unknown a priori. Some of the characteristics of this learning architecture are its sparsely connected units, the allocation of features and links in a data driven way, the decision mechanism and the utilization of an efficient update rule. In training the SNoW-based face detector, 1,681 face images from Olivetti [171], UMIST [65], Harvard [66], Yale [10] and FERET [141] databases are used to capture the variations in face patterns. To compare with other methods, they report results with two readily available data sets which contain 225 images with 619 faces [161]. With an error rate of 5.9%, this technique performs as well as other methods evaluated on the dataset 1 in [161], including those using neural networks [161], Kullback relative information [30], naive Bayes [174] and support vector machines [136], while being significantly more efficient computationally. See Table 2.4 for comparisons.

2.4.6 Bayesian Approach

In contrast to the methods in [187] [136] which model the global appearance of a face, Schneiderman and Kanade [174] describe a naive Bayes classifier based on local appearance and position of face patterns (subregions of the face) at multiple resolutions. They emphasize local appearance because some local patterns of an object are more unique than others; the intensity patterns around the eyes are much more distinctive than the pattern found around the cheeks. At each scale, a face image is decomposed into four rectangular subregions. These subregions are then projected to a lower dimensional space using PCA and quantized into a finite set of patterns, and the statistics of each projected subregion are estimated from the projected samples to encode local appearance. There are two reasons for adopting the naive Bayes assumption (i.e., no statistical dependency between the subregions). First, it provides better estimation of the conditional density functions of these subregions. Second, a naive classifier provides a functional form of the posterior probability to

capture the joint statistics of local appearance and position on the object. Under this formulation, their method decides that a face is present when the likelihood ratio is larger than the ratio of prior probabilities. With an error rate of 93.0% on data set 1 in [161], the proposed Bayesian approach shows comparable performance to [161] and is able to detect some rotated and profile faces. See Table 2.4 for comparisons.

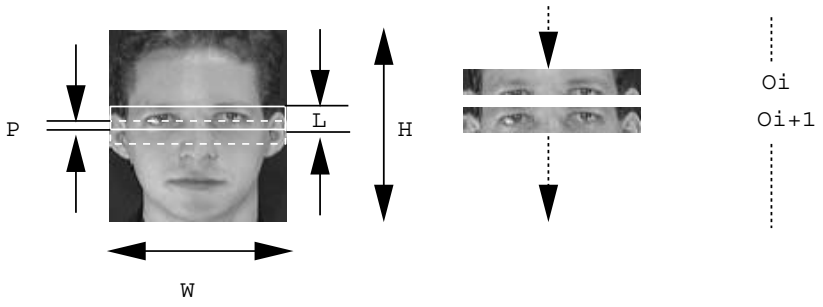
2.4.7 Hidden Markov Model

The underlying assumption of the Hidden Markov Model (HMM) is that patterns can be characterized as a parametric random process, and that the parameters of the stochastic process can be estimated in a precise, well-defined manner. In developing an HMM for a pattern recognition problem, a number of hidden states need to be decided first to form a model. Then, we can train HMM to learn the transitional probability between states from the examples where each example is represented as a sequence of observations. The goal of training an HMM is to maximize the probability of observing the training data by adjusting the parameters in an HMM model with the standard Viterbi segmentation method and Baum-Welch algorithms. After the HMM has been trained, the output probability of an observation determines the class to which it belongs.

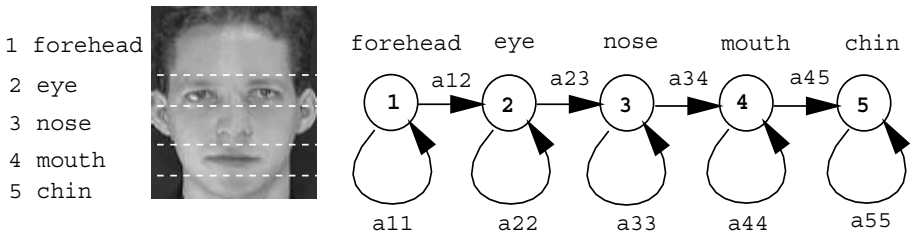
Intuitively a face can be divided into several regions such as the forehead, eyes, nose, mouth, and chin. Instead of relying on accurate alignment as in template matching or appearance-based methods (where facial features such as eyes and noses need to aligned well with respective to a reference point), this approach aims to associate facial regions with the states of a continuous density Hidden Markov Model. HMM-based methods usually treat a face pattern as a sequence of observation vectors where each vector is a strip of pixels as shown in Figure 2.7(a). During training and testing, an image is scanned in some order (usually from top to bottom) and an observation is taken as a block of pixels as shown in Figure 2.7(a).

For face patterns, the boundaries between strips of pixels are represented by probabilistic transitions between states as shown in Figure 2.7(b), and the image data within a region is modeled by a multivariate Gaussian distribution. An observation sequence consists of all intensity values from each block. The output states correspond to the classes to which the observations belong. After the HMM has been trained, the output probability of an observation determines the class to which it

belongs. HMMs have been applied to both face recognition and localization. Samaria [171] showed that the states of the HMM he trained corresponds to facial regions as shown in Figure 2.7(b). In other words, one state is responsible for characterizing the observation vectors of human foreheads, and another state is responsible for characterizing the observation vectors of human eyes. For face localization, an HMM is trained for a generic model of human faces from a large collection of face images. If the face likelihood obtained for each rectangular pattern in the image is above a threshold, a face is located.



(a)



(b)

Figure 2.7: Hidden Markov Model for face localization. (a) Observation vectors: To train an HMM, each face sample is converted to a sequence of observation vectors. Observation vectors are constructed from a window of $W \times L$ pixels. By scanning the window vertically with P pixels of overlap, an observation sequence is constructed. (b) Hidden states: When an HMM with five states is trained with sequences of observation vectors, the boundaries between states are shown in (b).

Samaria and Young apply 1D and pseudo 2-D HMMs to facial feature extraction and face recognition [171] [170]. Their HMMs exploit the structure of a face to enforce constraints on the state transitions. Since significant facial regions such as hair, forehead, eyes, nose and mouth occur in the natural oder from top to bottom, each of these regions is assigned to a state in a one-

dimensional continuous HMM. Figure 2.7(b) shows these five hidden states. For training, each image is uniformly segmented, from top to bottom into five states (i.e., each image is divided into five nonoverlapping regions of equal size). The uniform segmentation is then replaced by the Viterbi segmentation, and the parameters in the HMM are re-estimated using the Buam-Welch algorithm. As shown in Figure 2.7(a), each face image of width W and height H is divided into overlapping blocks of height L and width W. There are P rows of overlap between consecutive blocks in the vertical direction. These blocks form an observation sequence for the image, and the trained HMM is used to determine the output state. Similar to [170], Nefian and Hayes [133] apply HMMs and Karhunen Loève Transform (KLT) to face localization and recognition. Instead of using raw intensity values, the observation vectors consist of the (KLT) coefficients computed from the input vectors. Their experimental results on face recognition show a better recognition rate than [170]. On the MIT database which contains 432 images each with a single face, this pseudo 2-D HMM system has a success rate of 90%.

Rajagopalan et al. propose two probabilistic methods for face detection [153]. In contrast to [188] which uses a set of multivariate Gaussians to model the distribution of face patterns, the first method in [153] uses higher order statistics (HOS) for density estimation. Similar to [188], both the unknown distributions of faces and nonfaces are clustered using six density functions based on higher order statistics of the patterns. As in [187], a multilayer perceptron is used for classification, and the input vector consists of twelve distance measures (i.e., log probability) between the image pattern and the twelve model clusters. The second method in [153] uses an HMM to learn the face to nonface and nonface to face transitions in an image. This approach is based on generating an observation sequence from the image and learning the HMM parameters corresponding to this sequence. The observation sequence to be learned is first generated by computing the distance of the subimage to the centers of the twelve face and nonface cluster centers estimated in the first method. After the learning completes, the optimal state sequence is further processed for binary classification. Experimental results show that both HOS and HMM methods have a higher detection rate than [161] [188], but with more false alarms.

2.4.8 Information-Theoretical Approach

Lew applies Kullback relative information [33] to face detection by associating hypothesis H_1 to the event that the template is a face and H_0 to the event that the template is not a face [111]. A face training database consisting of 9 views of 100 individuals is used to estimate the face distribution. The nonface probability density function is estimated from a set of 143,000 nonface templates. From the training sets, the most informative pixels (MIP) are selected to maximize the Kullback relative information (i.e., to give the maximum class separation). It turns out the MIP distribution focuses on the eye and mouth regions and avoids the nose area. The MIP are then used to obtain linear features for classification and representation using the method of Fukunaga and Koontz [56]. To detect faces, a window is passed over the input image, and the distance from face space (DFFS) as defined in [139] is calculated. If the DFFS to the face subspace is lower than the distance to the nonface subspace, a face is assumed to exist within the window.

Kullback relative information is also employed by Colmenarez and Huang to maximize the discrimination between positive and negative examples of faces [29] [30]. They use a family of discrete Markov processes to model the face and background patterns and to estimate the probability model. The learning process is converted into an optimization problem to select the Markov process that maximizes the information-based discrimination between the two classes. The likelihood ratio is computed using the trained probability model and used to detect the faces.

Qian and Huang [150] present a method that employs the strategies of both view-based and model-based methods. First, a visual attention algorithm, which uses high level domain knowledge, is applied to reduce the search space. This is achieved by selecting image areas in which targets may appear based on the region maps generated by a region detection algorithm (water-shed method). Within the selected regions, faces are detected with a combination of template matching methods and feature matching using a hierarchical Markov random field and maximum *a posterior* estimation.

2.4.9 Inductive Learning

Inductive learning algorithms have also been applied to locate and detect faces. Huang, Gutta and Wechsler [77] apply Quinlan's C4.5 algorithm [152] to learn a decision tree from positive and

negative examples. Each training example is an 8×8 pixel window and is represented by a vector of 30 attributes which is composed of entropy, mean and standard deviation of the pixel intensity values. From these examples, C4.5 builds a classifier as a decision tree whose leaves indicate class identity and whose nodes specify tests to perform on a single attribute. The learned decision tree is then used to decide whether a face exists in the input example. The experiments show a localization accuracy rate of 96% on a set of 2,340 frontal face images in the FERET dataset.

Duta and Jain [42] present a method to learn the face concept using Mitchell's Find-S algorithm [129]. Similar to [188], they conjecture that the distribution of face patterns $p(\mathbf{x}|face)$ can be approximated by a set of Gaussian clusters, and that the distance from a face instance to one of the cluster centroids should be smaller than a fraction of the maximum distance from the points in that cluster to its centroid. The Find-S algorithm is then applied to learn the thresholding distance such that faces and nonfaces can be differentiated. This method has several distinct characteristics. First, it does not use negative (nonface) examples while [188] [161] use both positive and negative examples. Second, only the central portion of a face is used for training. Third, feature vectors consist of mosaic images with 32 intensity levels or textures while [188] uses full-scale intensity values as inputs (See Figure 2.1 for examples of mosaic images). This method achieves a detection rate of 85% on the CMU dataset.

2.5 Face Image Databases and Performance Evaluation

Most face detection methods require a training dataset of face images, and the databases originally developed for face recognition experiments can be used as training sets for face detection. Since these databases were constructed to empirically evaluate recognition algorithms in certain domains, we first review the characteristics of these databases and their applicability to face detection. Although numerous algorithms have been developed for face detection, most have not been tested on datasets with a large number of face images. Furthermore, most experimental results are reported using different test sets. In order to fairly compare methods, a few benchmark datasets have recently been compiled. We review these benchmark datasets and discuss their features. There are still a few issues that need to be carefully considered in performance evaluation even when the methods use the same test set. One issue is that researchers have different interpretations of what

a "successful detect" is. Another issue particularly for appearance-based methods is that different training sets are used We conclude this section with a discussion on these issues.

2.5.1 Face Image Database

Although many face detection methods have been proposed, less attention has been paid to the development of an image database for face detection and recognition research. The FERET [142] database, developed by the Army Research Laboratory with George Mason University, was obtained using a photographic camera. The database consists of monochrome images taken in different frontal views, and in left and right half profiles. Only the upper torso of an individual (mostly head and necks) appears in an image on a uniform and uncluttered background. The FERET database has been used to assess the strengthens and weaknesses of different face recognition approaches [140] [155]. Since each image consists of an individual on a uniform and uncluttered background, it is not suitable for face detection benchmarking. This is similar to many databases that were created for the development and testing of face recognition algorithms. Turk and Pentland [199] created a face database of 16 people (available at ftp://whitechapel.media.mit.edu/pub/images/). The images are taken in frontal view with slight variability in head orientation (tilted upright, right and left) on a cluttered background. The face database from AT&T Cambridge Laboratories (formerly known as the Olivetti database) [171] consists of ten different images for forty distinct subjects, and is available at http://www.uk.research.att.com/facedatabase.html. The images were taken at different times, varying the lighting, facial expressions and facial details (glasses). The Harvard database [66] consists of cropped, masked frontal face images taken from a wide variety of light sources. It was used by Hallinan for a study on face recognition under the effect of varying illumination conditions. With sixteen individuals, the Yale face database [10] (available at http://cvc.yale.edu/) contains ten frontal images per person, each with different facial expressions, with and without glasses, and under different lighting conditions. The M2VTS multimodal database [143] from the European ACTS projects was developed for access control experiments using multimodal inputs. It contains sequences of face images of 37 people. The five sequences for each subject were taken over one week. Each image sequence contains images from right profile (-90 degree) to left profile (90 degree) while the subject counts from '0' to '9' in their native languages. Sequences are thus

generated for each subject: the voice sequence, the motion sequence and the glasses off motion sequence (if any). This database has been applied to face localization [181], face recognition and face authentication [193] [103]. The UMIST database [65], compiled by Graham and Allinson, consists of 564 images of 20 people with varying pose. The images of each subject cover a range of poses from right profile to frontal views. The Purdue AR database [122], recently created by Martinez and Benavente, contains over 3,276 color images of 126 people (70 males and 56 females) in frontal view. This database is designed for face recognition experiments under several mixing factors such as facial expressions, illumination conditions and occlusions. All the faces appear with different facial expression (neutral, smile, anger, and scream), illumination (left light source, right light source and sources from both sides), and occlusion (wearing sunglasses or scarf). The images were taken during two sessions separated by two weeks. All the images were taken by the same camera setup under tightly controlled conditions of illumination and pose. This face database has been applied to image and video indexing as well as retrieval [122]. Table 2.2 summarizes the characteristics of some face image databases.

2.5.2 Benchmark Test Sets for Face Detection

The abovementioned databases are designed mainly for testing face recognition methods, and thus each image contains only one individual. Therefore, such databases can be best utilized as training rather than test samples. Sung and Poggio [187] [188] created two databases for face detection. The first set consists of 301 frontal and near-frontal mugshots of 71 different people. These images are high quality digitized images with a fair amount of lighting variation. The second set consists of 23 images with a total of 149 face patterns. Most of these images have complex background with faces taking up only a small amount of the total image area. The most widely-used face detection database has been created by Rowley, Baluja, and Kanade [160] [163] (available at http://www.cs.cmu.edu/~har/faces.html). It consists of 130 images with a total of 507 frontal faces, including 23 images of the second data set used by Sung and Poggio [188]. Most images contain more than one face on a cluttered background, and so this is a good test set to assess algorithms which detect upright frontal faces. Figure 2.8 shows some images in the dataset collected by Sung and Poggio [188], and Figure 2.9 shows images from the dataset collected by Rowley, Baluja, and

Table 2.2: Face image database

Dataset	Location	Description
MIT face Database [199]	ftp://whitechapel.media.mit.edu /pub/images	Faces of 16 people, 27 of each person under various illumination conditions, scale and head orientation.
FERET	National Institute of Standards and Technology	A large collection of male and female faces. Each image contains a single person with certain expression.
UMIST [65]	http://images.ee.umist.ac.uk/danny/ database.html	Contains 564 images of 20 subjects. Each subject covers a range of poses from profile to frontal views.
University of Bern	ftp://iamftp.unibe.ch /pub/Images/FaceImages/	Contains 300 frontal view face images of 30 people (10 images per person) and 150 profile face images (5 images per person).
Yale Database [10]	http://cvc.yale.edu	Contains face images with expressions, glasses under different illumination conditions.
AT&T (Olivetti) Database [171]	http://www.uk.research.att.com	Consists of 40 subjects, 10 images per subject.
Harvard Database [66]	ftp://ftp.hrl.harvard.edu/pub/faces/	Contains cropped, masked frontal face images under a wide range of illumination conditions.
M2VTS Database [143]	http://poseidon.csd.auth.gr /M2VTS/index.html	A multimodal database containing various image sequences of different persons.
Purdue AR Database [122]	http://rvl1.ecn.purdue.edu/ ~aleix/aleix_face _DB.html	Contains 3,276 face images with different facial expressions and occlusions under different illuminations.

Kanade [161].

Figure 2.8: Sample images in Sung's dataset. Some of these images are scanned from newspapers and thus have low resolution. Though most faces in the images are upright and frontal. Some faces in the images appear in different pose.

Rowley, Baluja, and Kanade [162] also compiled another database of images for detecting rotated 2-D faces. It contains 50 images with a total of 223 faces, of which 210 are at angles of more than 10 degrees. Figure 2.10 shows some rotated images in this data set.

Recently, Kodak compiled an image database [118] as a common testbed for direct benchmarking of face detection and recognition algorithms. Their database has 300 digital photos that are captured in a variety of resolutions and face size ranges from as small as 13×13 pixels to as large as 300×300 pixels.

Table 2.3 summarizes the characteristics of the abovementioned test sets for face detection.

Table 2.3: Test sets for face detection

Dataset	Location	Description
Sung's Dataset [188]	http://www.cs.cmu.edu/~har	Contains two sets of high and low resolution intensity images that contains multiple faces in complex background.
CMU Datasets [161]	http://www.cs.cmu.edu/~har	Consists of 130 intensity images with a total 507 frontal faces and it is a superset of Sung's dataset.
Kodak Dataset [118]	Eastman Kodak Corporation	Contains faces of multiple size, pose and under varying illumination in color images. Designed as a testbed for face detection and recognition.

Figure 2.9: Sample images in Rowley's dataset. Some images contain hand-drawn cartoon faces. Most images contain more than one faces. Also, the face size varies.

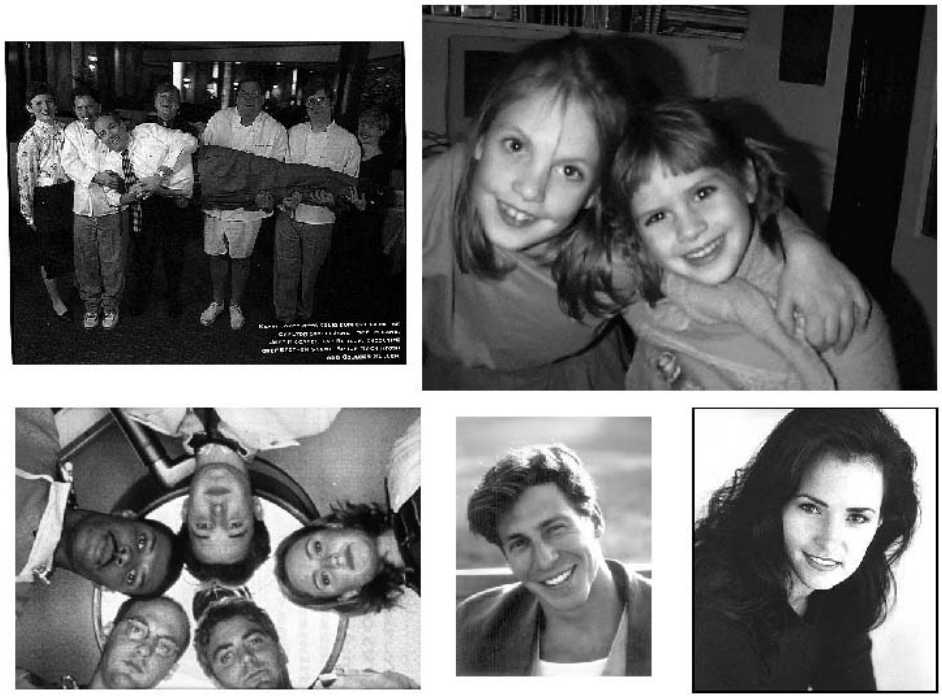


Figure 2.10: Sample images of non-upright faces from Rowley's dataset. This dataset contains faces in different orientation and some faces in different pose.

2.5.3 Performance Evaluation

In order to have a fair empirical evaluation on face detection methods, it is important to use a standard and representative dataset for experiments. Although many face detection methods have been developed for the past decade, only a few of them have been tested on the same dataset. Table 2.4 summarizes the reported performance among several appearance-based face detection methods on two standard datasets described in the previous section.

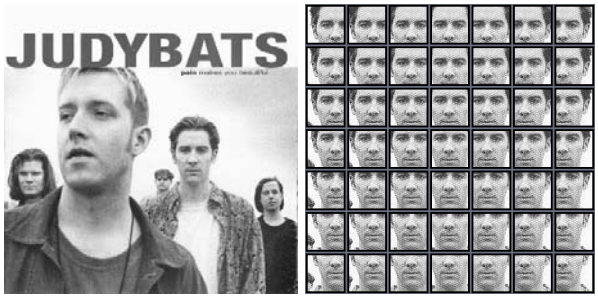
Although Table 2.4 shows the performance of these methods on the same dataset, it is still difficult to have a fair evaluation. There are at least three factors that complicate the assessment of these appearance-based methods. First, these learning methods usually use different training sets and different tuning parameters. Therefore, the number of training examples has a direct effect on the classification performance. However this factor is often ignored in evaluating methods, which is a good criteria if the goal is to evaluate the systems rather than the learning methods. The second factor is the training time and execution time. Although the training time is usually ignored by most systems, it may be important for real-time applications that require training on

Table 2.4: Experimental results on images from test set 1 (125 images with 483 faces) and test set 2 (23 images with 136 faces) (see text for details)

Method	Test Set 1		Test Set 2	
	Detect Rate	False Detects	Detect Rate	False Detects
(1) Distribution-based [188]	N/A	N/A	81.9%	13
(2) Neural network [161]	92.5%	862	90.3%	42
(3) Naive Bayes [174]	93.0%	88	91.2%	12
(4) Kullback relative information [30]	98.0%	12758	N/A	N/A
(5) Support vector machine [136]	N/A	N/A	74.2%	20
(6) Mixture of factor analyzers [222]	92.3%	82	89.4%	3
(7) Fisher linear discriminant [223]	93.6%	74	91.5%	1
(8) SNoW with primitive features [226]	94.2%	84	93.6%	3
(9) SNoW with multi-scale features [226]	94.8%	78	94.1%	3

different datasets. Finally, the number of scanning windows in these methods vary because they are designed to operate in different environments (i.e., to detect faces within a size range). For example, Colmenarez and Huang [30] argue that their method scans more windows than others and thus the number of false detects is higher than others. Furthermore, the criteria adopted in reporting the detect rates is usually not clearly described in most systems. Figure 2.11(a) shows a test image and Figure 2.11(b) shows some scanning windows to be classified. Suppose all the subimages in Figure 2.11(b) are classified as face patterns by a classifier, a loose criterion may consider all of them as "successful" detects. However, a more strict criterion (e.g., each successful detect must contain all the visible eyes and mouthes in an image) will classify most of them as false alarms. It is clear that a uniform criteria should be adopted to assess different classifiers.

In [161] Rowley, Baluja, and Kanade adjust the criteria until the experimental results match their intuition of what a correct detection is, i.e., the square window should contain the eyes and also the mouth. The criteria they eventually use is that the center of the detected bounding box must be within 4 pixels and the scale must he within a factor of 1.2 (their scale step size) of ground truth (recorded manually). On the other hand in [222] [223] [226], a face is considered correctly detected if the bounding box of the detected face contains at least one eye and the mouth. To make the comparison more difficult, the criteria may and should depend on the purpose of the detector. For instance, one detector may attempt to locate the eyes and upper face only. In this case, a successful detect may contain only 40% of the whole face. But for a detector designed for face recognition or facial expression recognition, such a detect should be counted as an error.



(a) Test image (b) Detection results

Figure 2.11: Different criteria lead to different detection results. Suppose all the subimages in (b) are classified as face patterns by a classifier. A loose criterion may declare all the faces as "successful" detects while a more strict one would declare most of them as nonfaces.

2.6 Discussion and Conclusion

Although significant progress has been made in the last two decades, there is still work to be done, and we believe that a robust face detection system should be effective under full variation in :

- lighting conditions
- orientation, pose, and partial occlusion
- facial expression
- presence of glasses, facial hair, variety of hair styles

In addition, there is a need to create a convincing evaluation methodology and representative databases for benchmarking.

Face detection is a challenging and interesting problem in and of itself. However, it can also be seen as a one of the few attempts at solving one of the grand challenges of computer vision, the recognition of object classes. The class of faces admits a great deal of shape, color, and albedo variability due to differences in individuals, non-rigidity, facial hair, glasses, and makeup. Images are formed under variable lighting and 3-D pose, and may have cluttered backgrounds. Hence, face detection research confronts the full range of challenges found in general purpose, object class recognition. However, the class of faces also has very apparent regularities that are exploited by many heuristic or model-based methods or are readily "learned" in data-driven methods. One

expects some regularities when defining classes in general, but they may not be so apparent. Finally, though faces have tremendous within-class variability, face detection remains a two class recognition problem (face vs. non-face).

Face detection also has practical importance since it is the first step in any fully automated face recognition or face tracking system. With the growing interest in human-computer interactions, it is important to investigate fast and robust methods. We have not addressed the issue of speed or computational complexity here, since the aim of most face detection work is on accuracy, not speed. This section gives a critical review of existing methods and discusses several promising directions for future research.

Chapter 3

Recognizing Hand Gestures Using Motion Trajectories

We present an algorithm for extracting and classifying two-dimensional motion in an image sequence based on motion trajectories. First, a multiscale segmentation is performed to generate homogeneous regions in each frame. Regions between consecutive frames are then matched to obtain 2-view correspondences. Affine transformations are computed from each pair of corresponding regions to define pixel matches. Pixels matches over consecutive images pairs are concatenated to obtain pixel-level motion trajectories across the image sequence. Motion patterns are learned from the extracted trajectories using a time-delay neural network. We apply the proposed method to recognize 40 hand gestures of American Sign Language. Experimental results show that motion patterns in hand gestures can be extracted [217] and recognized with high recognition rate using motion trajectories [218] [220].

3.1 Introduction

In this chapter, we present an algorithm for extracting two-dimensional motion fields of objects across a video sequence and classifying each as one of a set of *a priori* known classes. The algorithm is used to recognize dynamic visual processes based on spatial, photometric and temporal characteristics. An application of the algorithm is in sign language recognition where an utterance is interpreted based on, for example, hand location, shape, and motion. The performance of the algorithm is evaluated on the task of recognizing 40 complex hand gestures of American Sign Language (ASL).

The algorithm consists of two major steps. First, each image is partitioned into regions using a multiscale segmentation method. Regions between consecutive frames are then matched to obtain 2-view correspondences. Affine transformations are computed from each pair of corresponding regions to define pixel matches. Pixel matches over consecutive image pairs are concatenated to obtain pixel-level motion trajectories across the video sequence. Pixels are also grouped based on their 2-view motion similarity to obtain a motion based segmentation of the video sequence. Only some of the moving regions correspond to visual phenomena of interest. Both the intrinsic properties of the objects represented by image regions and their dynamics represented by the motion trajectories determine whether they comprise an event of interest. For example, it is sufficient to recognize most gestures in ASL in terms of shape and location changes of palm regions. Therefore, palm and head regions are extracted out in each frame and the palm locations are specified with reference to the usually still head regions.

To recognize motion patterns from trajectories, we use a time-delay neural network (TDNN) [206]. TDNN is a multilayer feedforward network that uses time-delays between all layers to represent temporal relationships between events in time. An input vector is organized as a temporal sequence, where only the portion of the input sequence within a time window is fed to the network at one time. The time window is shifted and another portion of the input sequence is given to the network until the whole sequence has been scanned through. The TDNN is trained using standard error backpropagation learning algorithm. The output of the network is computed by adding all of these scores over time, followed by applying a nonlinear function such as sigmoid function to the sum. TDNNs with two hidden layers using sliding input windows over time lead to a relatively small number of trainable parameters. We adopt TDNN to recognize motion patterns because gestures are spatio-temporal sequences of feature vectors defined along motion trajectories. Our experimental results show that motion patterns can be learned by a time-delay neural network with high recognition rate.

3.2 Related Work

Since Johansson's seminal work [83] that suggests human movements can be recognized solely by motion information, motion profiles and trajectories have been investigated to recognize human

motion by several researchers. In [179] Siskind and Morris conjecture that human event perception does not presuppose object recognition. In other words, they think visual event recognition is performed by a visual pathway which is separated from object recognition. To verify the conjecture, they analyze motion profiles of objects that participate in different simple spatial-motion events. Their tracker uses a mixture of color based and motion based techniques. Color based techniques are used to track objects defined by set of colored pixels whose saturation and value are above certain thresholds in each frame. These pixels are then clustered into regions using a histogram based on hue. Moving pixels are extracted from frame differences and divided into clusters based on proximity. Next, each region (generated by color or motion) in each frame is abstracted by an ellipse. Finally, feature vector for each frame is generated by computing the absolute and relative ellipse positions, orientations, velocities and accelerations. To classify visual events, they use a set of Hidden Markov Models (HMMs) which are used as generative models and trained on movies of each visual event represented by a set of feature vectors. After training, a new observation is classified as being generated by the model that assigns the highest likelihood. Experiments on a set of 6 simple gestures, "pick up," "put down," "push," "pull," "drop," and "throw," demonstrate that gestures can be classified based on motion profiles.

Bobick and Wilson [209] adopt a state based approach to represent and recognize gestures. First, many samples of a gesture are used to compute its principal curve [69] which is parameterized by arc length. A by-product of calculating the curve is the mapping of each sample point of a gesture example to an arc length along the curve. Next, they use line segments of uniform length to approximate the discretized curve. Each line segment is represented by a vector and all the line segments are grouped into a number of clusters. A state is defined to indicate the cluster to which a line segment belongs. A gesture is then defined by an ordered sequence of states. The recognition procedure is to evaluate whether input trajectory successfully passes through the states in the prescribed order. Contrasted to their work where each example of a gesture is a single trajectory in space, each gesture in our work is represented by a set of motion trajectories corresponding to the motions of different parts of, say, the palm, instead of a single representative point. Thus, each example of a gesture in our work is represented by a set of motion trajectories. Our experimental results show that an ensemble of trajectories yields better generalization

Recently, Isard and Blake have proposed the CONDENSATION algorithm [79] as a probabilistic method to track curves in visual scenes. This method is a fusion of the statistical factored sampling algorithm with a stochastic model to search a multivariate parameter space that is changing over time. Objects are modeled as a set of parameterized curves and the stochastic model is estimated based on the training sequence. Experiments on the proposed algorithm have been carried to track objects based on their hand drawn templates. Black and Jepson [12] extend this algorithm to recognize gestures and facial expressions in which human motions are modeled as temporal trajectories of some estimated parameters (which describe the states of a gesture) over time. The major difference between our approach and these methods is that we propose a method to extract motion trajectories from an image sequence without hand drawn templates [79] or distinct trackable icons [12]. Motion patterns are then learned from the extracted motion trajectories. No prior knowledge is assumed or required for the extraction of motion trajectories, although domain specific knowledge can be applied for efficiency reasons.

3.3 Motion Segmentation

To capture the dynamic characteristics of objects, we segment an image frame into regions with uniform motion. Our motion segmentation algorithm processes an image sequence two successive frames at a time. For a pair of frames, (I_t, I_{t+1}), the algorithm identifies regions in each frame comprising the multiscale intraframe structure. Regions at all scales are then matched across frames. Affine transforms are computed for each matched region pair. The affine transform parameters for region at all scales are then used to derive a single motion field which is then segmented to identify the differently moving regions between the two frames. The following sections describe the major steps in the motion segmentation algorithm.

3.3.1 Multiscale Image Segmentation

Multiscale segmentation is performed using a transform descried in [2] which extracts a hierarchy of regions in each image. The general form of the transform, which maps an image to a family of

attraction force fields, is defined by

$$\mathbf{F}(x,y;\sigma_g(x,y),\sigma_s(x,y)) = \int\int_R d_g(\Delta I,\sigma_g(x,y))\cdot d_s(\vec{r},\sigma_s(x,y))\frac{\vec{r}}{||\vec{r}||}dwdv$$

where $R = domain(I(u,v))\backslash\{(x,y)\}$ and $\vec{r} = (v-x)\vec{i} + (w-y)\vec{j}$. The parameter σ_g denotes a homogeneity scale which reflects the homogeneity of a region to which a pixel belongs and σ_s is spatial scale that controls the neighborhood from which the force on the pixel is computed. The homogeneity of two pixels is given by the Euclidean distance between the associated m-dimensional vectors of pixel values (e.g., $m = 3$ for a color image):

$$\Delta I = |I(x,y) - I(v,w)|$$

The spatial scale parameter, σ_s, controls the spatial distance function, $d_s(\cdot)$, and the homogeneity scale parameter, σ_g, controls the homogeneity distance function, $d_g(\cdot)$. One possible form for these functions satisfying criteria discussed in [2] is unnormalized Gaussian:

$$\begin{aligned} d_g(\Delta I,\sigma_g) &\sim \sqrt{2\pi\sigma_g^2}N_{\Delta_I}(0,\sigma_g^2) \\ d_s(\vec{r},\sigma_s) &\sim \begin{cases} \sqrt{2\pi\sigma_s^2}N_{||\vec{r}||}(0,\sigma_s^2), & ||\vec{r}|| \le 2\sigma_s \\ 0, & ||\vec{r}|| > 2\sigma_s \end{cases} \end{aligned}$$

The force field encodes the region structure in a manner which allows easy extraction. Region boundaries correspond to diverging force vectors in $\mathbf{F}$ and region skeletons correspond to converging force vectors in $\mathbf{F}$. An increase in σ_g causes less homogeneous structures to be encoded and an increase in σ_s causes large structures to be encoded.

3.3.2 Region Matching

The matching of motion regions across frames is formulated as a graph matching problem at four different scales where scale refers to the level of detail captured by the image segmentation process. Three partitions of each image are created by slicing through the multiscale pyramid at three preselected values of σ_g. Region partitions from adjacent frames are matched from coarse to fine scales, with coarser scale matches guiding the finer scale matching. Each partition is represented as

a region adjacency graph, within which each region is represented as a node and region adjacencies are represented as edges. Region matching at each scale consists of finding the set of graph transformation operations (edge deletion, edge and node matching, and node merging) of least cost that create an isomorphism between the current graph pair. The cost of matching a pair of regions takes into account their similarity with regard to area, average intensity, expected position as estimated from each region's motion in previous frames, and the spatial relationship of each region with its neighboring regions.

Once the image partitions at the three different homogeneity scales have been matched, matchings are then obtained for the regions in the first frame of the frame pair that were identified by the motion segmentation module using the previous frame pair. The match in the second frame for each of these motion regions is given as the union of the set of finest scale regions that comprise the motion region. This gives a fourth matched pair of image partitions, and is considered to be the coarsest scale set of matches that is utilized in affine estimation. The details of the algorithm can be found in [191].

3.3.3 Affine Transformation Estimation

For each pair of matched regions, the best affine transformation between them is estimated iteratively. Let R_i^t be the ith region in frame t and its matched region be R_i^{t+1}. Also let the coordinates of the pixels within R_i^t be (x_{ij}^t, y_{ij}^t), with $j = 1 \ldots |R_i^t|$ where $|R_i^t|$ is the cardinality of R_i^t, and the pixel nearest the centroid of R_i^t be $(\bar{x}_i^t, \bar{y}_i^t)$. Each (x_{ij}^t, y_{ij}^t) is mapped by an affine transformation to the point $(\hat{x}_{ij}^t, \hat{y}_{ij}^t)$ according to

$$\begin{aligned}\begin{pmatrix} x_{ij}^t \\ y_{ij}^t \end{pmatrix} &\rightarrow R\left[\mathbf{A}_k \begin{pmatrix} x_{ij}^t - \bar{x}_i^t \\ y_{ij}^t - \bar{y}_i^t \end{pmatrix} + \vec{T}_k + \begin{pmatrix} \bar{x}_i{}^{t+1} \\ \bar{y}_i{}^{t+1} \end{pmatrix} \right] \\ &= \begin{pmatrix} \hat{x}_{ij}^t \\ \hat{y}_{ij}^t \end{pmatrix}_k\end{aligned}$$

where the subscript k denotes the iteration number, and $R[\cdot]$ denotes a vector operator that rounds each vector component to the nearest integer. The affine transformation comprises a 2×2 defor-

mation matrix, $\mathbf{A}_k$, and a translation vector, $\vec{T}_k$. By defining the indicator function,

$$\lambda_i^t(x,y) = \begin{cases} 1, (x,y) \in R_i^t \\ 0, else \end{cases}$$

the amount of mismatch is measured as

$$(M_i^t) = \sum_{x,y} |I_t(x,y) - I_{t+1}(\hat{x},\hat{y})| \cdot$$
$$\left[\lambda_i^t(x,y) + \lambda_i^{t+1}(\hat{x},\hat{y}) - \lambda_i^t(x,y) \cdot \lambda_i^{t+1}(\hat{x},\hat{y}) \right]$$

The affine transformation parameters that minimize M_i^t are estimated iteratively using a local descent criterion.

3.3.4 Motion Field Integration

The computed affine parameters give a motion field at each of the four scales. These motion fields are then combined into a single motion field by taking the coarsest motion field and then performing the following computation recursively at four scales. At each matched region, the image prediction error generated by the current motion field and the motion field at the next finer scale are compared. At any region where the prediction error using the finer scale motion improves by a significant amount, the current motion is replaced by the finer scale motion. The result is a set of "best matched" regions at the coarsest acceptable scales.

3.3.5 Motion Field Segmentation

The resulting motion field $\vec{M}_{t,t+1}$ is segmented into areas of uniform motion. We use a heuristic that considers each pair of best matched regions, R_i^t and R_j^t, which share a common border, and merges them if the following relation is satisfied for all (x_{ik}^t, y_{ik}^t) and (x_{jl}^t, y_{jl}^t) that are spatially adjacent to one another:

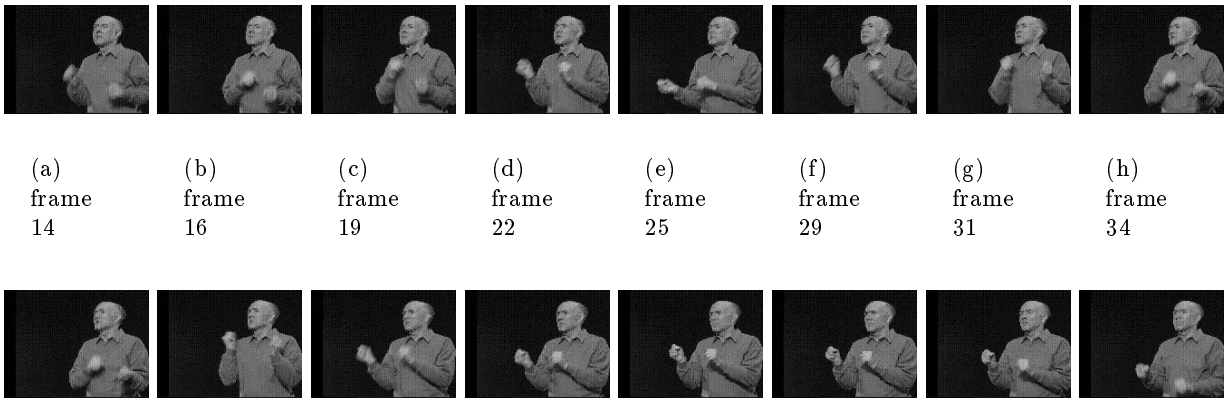
$$\frac{||\vec{M}_{t,t+1}(x_{ik}^t, y_{ik}^t) - \vec{M}_{t,t+1}(x_{jl}^t, y_{jl}^t)||}{max(||\vec{M}_{t,t+1}(x_{ik}^t, y_{ik}^t)||, ||\vec{M}_{t,t+1}(x_{jl}^t, y_{jl}^t)||)} < m_{\sigma_g}$$

where m_{σ_g} is a constant less than 1 that determines the degree of motion similarity necessary for the regions to merge.

The segmented motion regions are each represented in $MS_{t,t+1}$ by a different value. Because each of the best matched regions have matches, the matches in frame $t+1$ of the regions in $MS_{t,t+1}$ are known and comprise the coarsest scale regions that are used in the affine estimation module for the next frame pair.

It should be noted that the motion segmentation does not necessarily correspond to the moving objects in the scene because the motion segmentation is done over a single motion field. Nonrigid objects, such as humans, are segmented into multiple, piecewise rigid regions. In addition, fast objects moving at rates less than one pixel per frame cannot be identified. Handling both these situations requires examining the motion field over multiple frames.

Figure 3.1 shows frames from an image sequence of a complex ASL sign called "cheerleader" and Figure 3.2 shows the results of motion segmentation. Different motion regions are displayed with different gray levels. Notice that there are several motion regions within the head and palm regions because these piecewise rigid regions have uniform motion.



(a) frame 14 (b) frame 16 (c) frame 19 (d) frame 22 (e) frame 25 (f) frame 29 (g) frame 31 (h) frame 34

(i) frame 35 (j) frame 37 (k) frame 40 (l) frame 44 (m) frame 46 (n) frame 49 (o) frame 52 (p) frame 55

Figure 3.1: Image sequence of ASL sign "cheerleader"

Figure 3.5 shows frames from an image sequence of ASL sign "any" and Figure 3.8 shows another images from an ASL sign "anything." Figure 3.6 shows the results of motion segmentation. Different

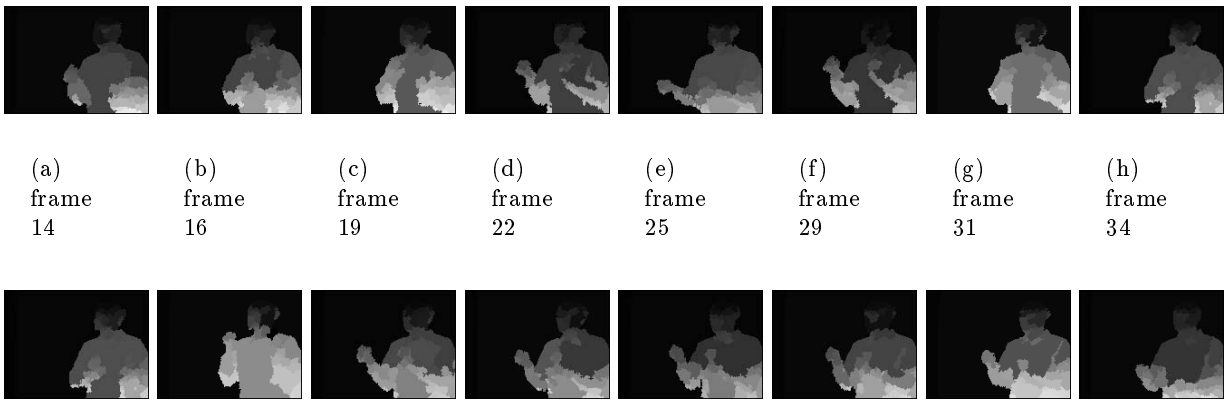


Figure 3.2: Motion segmentation of the image sequence "cheerleader" (pixels of the same motion region are displayed with same gray level and different regions are displayed with different gray levels)

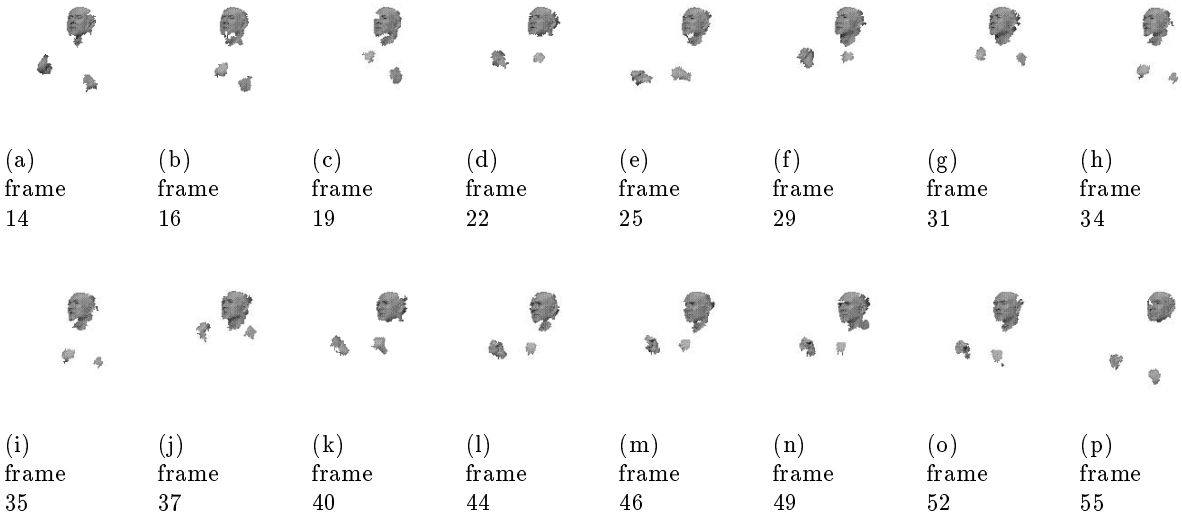
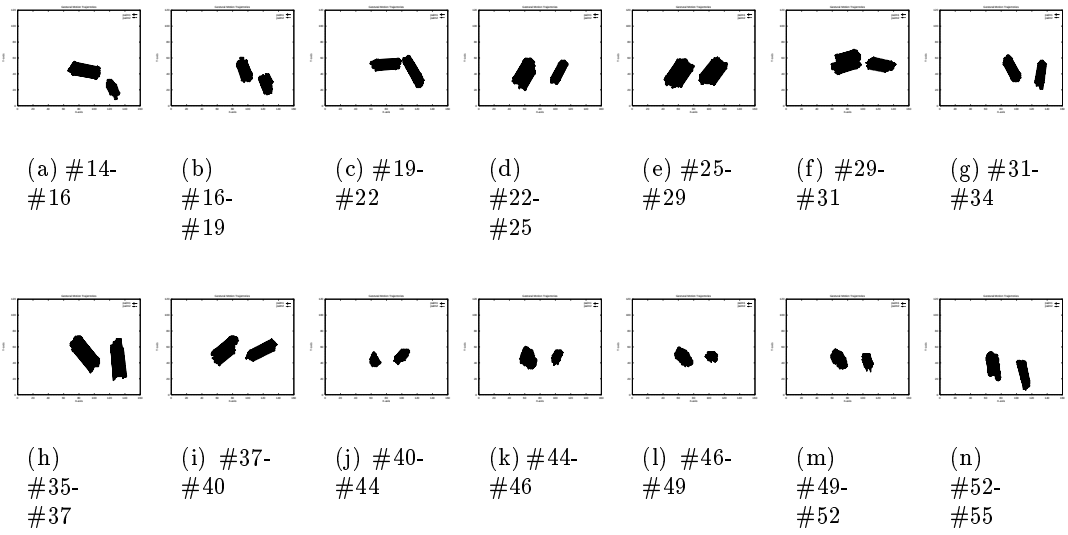


Figure 3.3: Extracted head and palm regions from image sequence "cheerleader"



(a) #14-#16 (b) #16-#19 (c) #19-#22 (d) #22-#25 (e) #25-#29 (f) #29-#31 (g) #31-#34

(h) #35-#37 (i) #37-#40 (j) #40-#44 (k) #44-#46 (l) #46-#49 (m) #49-#52 (n) #52-#55

Figure 3.4: Extracted gestural motion trajectories from segments of ASL sign "cheerleader" (since all pixel trajectories are shown, they form a thick blob)

motion regions are displayed with different gray levels. Notice that there are several motion regions within the head and palm regions because these piecewise rigid regions have similar motion.

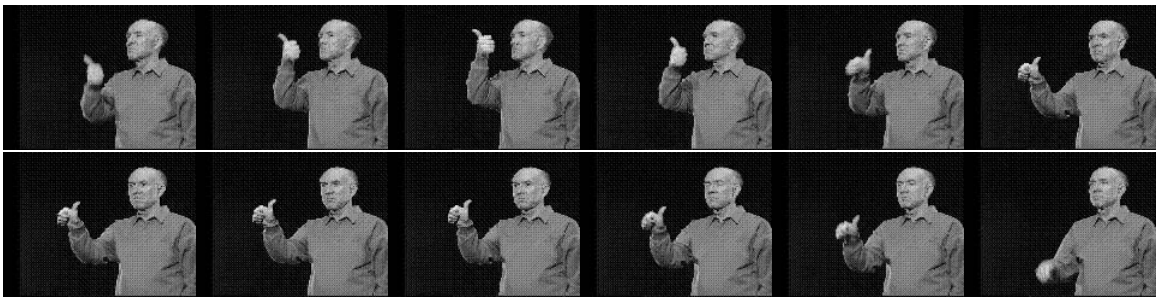


Figure 3.5: Image sequence of ASL sign "any" (time increases left to right and top to bottom)

3.4 Color and Geometric Analysis

Motion segmentation generates regions that have uniform motion. However, only some of these motion regions carry important information for motion pattern recognition. To recognize hand gestures considered here, it is sufficient to extract the motion regions of head and palm regions. Towards this end, we use color and geometric information of palm and head regions.

Human skin color has been used and proved to be an effective feature in many applications. We

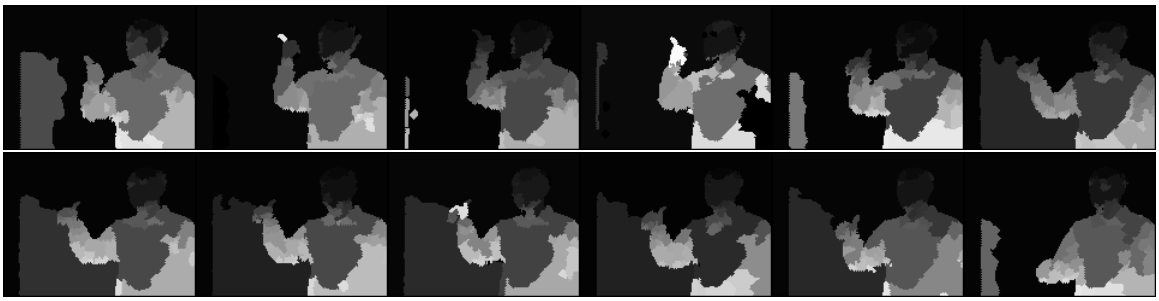


Figure 3.6: Motion segmentation of the sequence in Figure 3.5 (time increases left to right and top to bottom)

Figure 3.7: Extracted human head and palm regions in the sequence of Figure 3.5

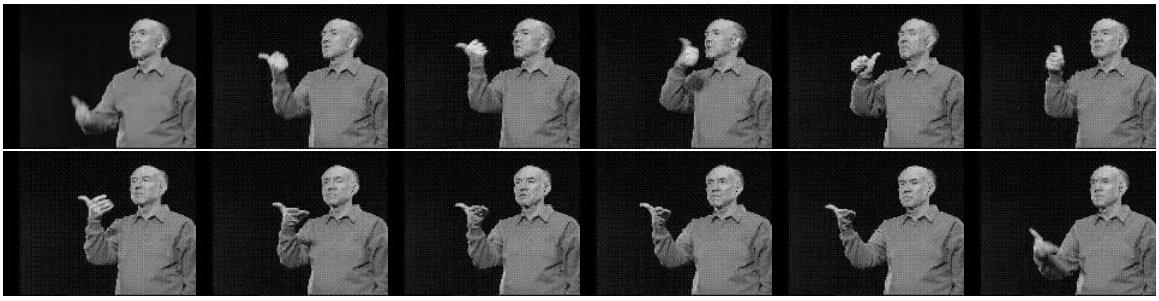


Figure 3.8: Image sequence of ASL sign "anything" (time increases left to right and top to bottom)

use a Gaussian mixture to model the distribution of skin color pixels from a Michigan database of 2,447 images which consists of human faces from different ethnic groups. We use CIE LUV color space and discard the luminescence value of each pixel to minimize the effects of lighting condition. The parameters in the Gaussian mixture are estimated using an EM algorithm. A motion region is classified to have skin color if most of the pixels have probabilities of being skin color above a threshold. Coupled with motion segmentation, motion regions of skin color can be efficiently extracted from image sequences.

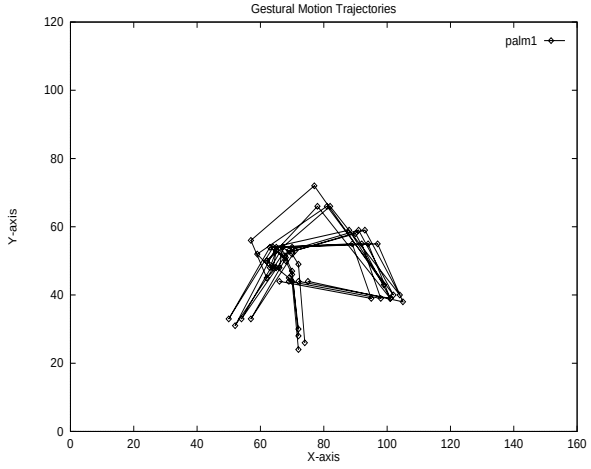
Since the shape of human head and palm can be approximated by ellipses, and the human hand is a thin rectangular region, motion regions that have skin color are merged until the shape of the merged region is approximately elliptic or rectangular. The parameters of a rectangular shape can be obtained from the bounding box of each region easily. The orientation of an ellipse is calculated from the axes of the least moment of inertia. The extents of the major and minor axes of the ellipse are approximated by the extents of the region along the axis directions, and thus generate the parameters for the ellipse. The largest elliptic region extracted from an image is identified as human head and the next two smaller elliptic regions are palm regions. Figure 3.1 shows the image sequence of a complex ASL sign called "cheerleader" and Figure 3.3 shows the results of color and geometric analysis on the motion regions.

3.5 Motion Trajectories

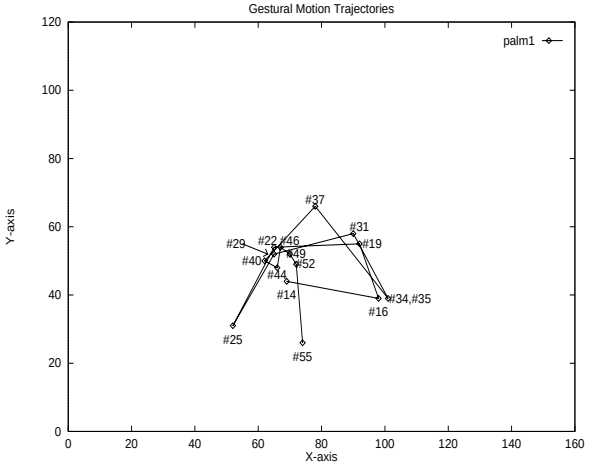
Although motion segmentation generates affine transformations that capture motion details by matching regions at fine scales, it is sufficient to use coarser motion trajectories of identified palm regions for gesture recognition considered in this work.

Affine transformation of palm region in each frame pair is computed based on equations in Section 3.3.3. The affine transformations of successive pairs are then concatenated to construct the motion trajectories of the palm region. Figure 3.4 shows such trajectories for a number of frames in the image sequence "cheerleader." Since all pixel trajectories are shown together, they form a thick blob. Figure 3.9 shows a 10 to 1 subsampling of the motion trajectories.

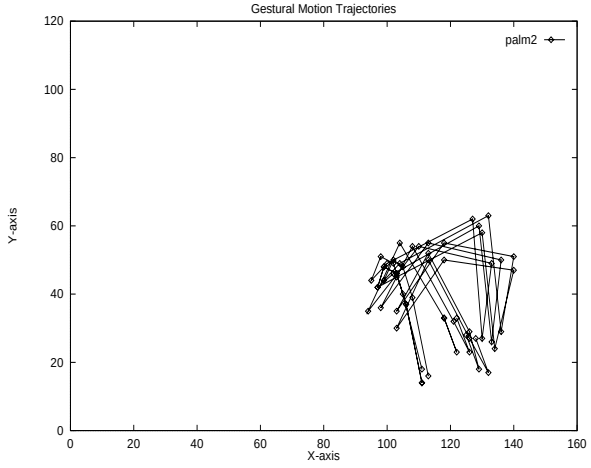
Motion regions with skin color are identified because of their chromatic characteristics. These regions are then merged into palm and head regions based geometric analysis as shown in Figure



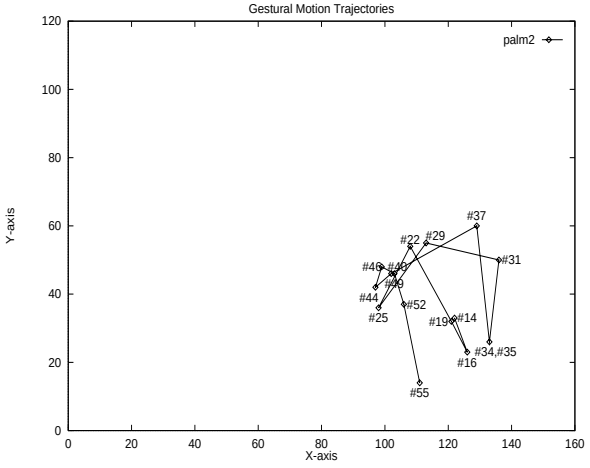
(a) Motion trajectories of a sample set of palm points for the ASL sign "cheerleader"



(b) Motion trajectory of one palm point for the ASL sign "cheerleader"



(c) Motion trajectories of a sample set of palm points for the ASL sign "cheerleader"



(d) Motion trajectory of one palm point for the ASL sign "cheerleader"

Figure 3.9: Extracted gestural motion trajectories (subsampled by a factor of 10) of ASL sign "cheerleader"

3.7. Affine parameters of matched palm regions are computed, thereby yielding motion trajectories. Figures 3.10 and 3.11 show the extracted motion trajectories from the image sequences of ASL sign "any" and "anything." Note that the motion trajectories of palm region matches the movement in the real scene well.

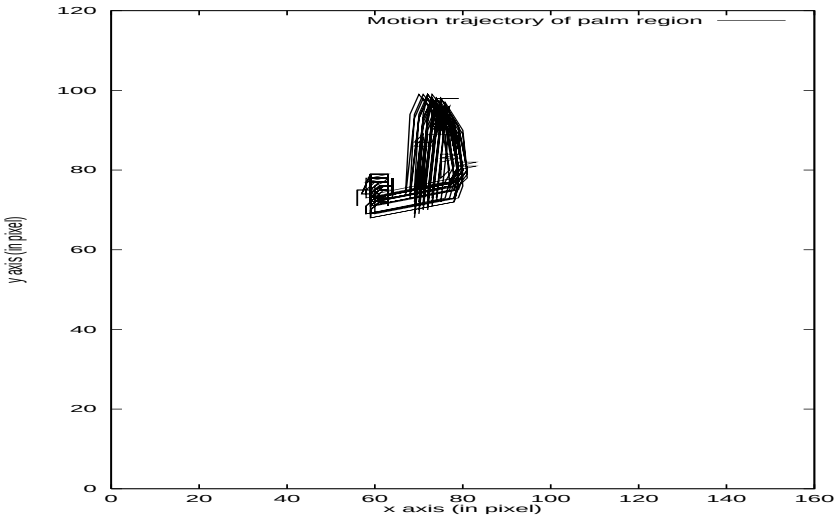


Figure 3.10: Motion patterns of gesture "any"

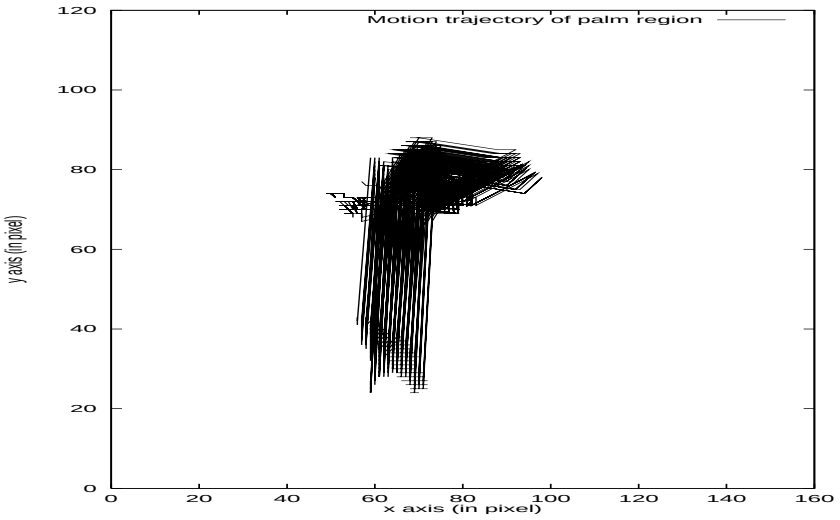


Figure 3.11: Motion patterns of gesture "anything"

3.6 Motion Pattern Classification

We employ TDNN to classify gestural motion patterns of palm regions since TDNNs have been demonstrated to be very successful in learning spatio-temporal patterns. TDNN is a dynamic classification approach in that the network sees only a small window of the motion pattern and this window slides over the input data while the network makes a series of local decisions. These local decisions have to be integrated into a global decision at a later time. In their seminal work, Waibel et al. [206] demonstrated excellent results for phoneme classification using TDNN and showed that it achieves lower error rates than those achieved by a simple HMM recognizer.

The design of TDNN is attractive because its compact structure economizes on weights and makes it possible for the network to develop general feature detectors. Most importantly, its temporal integration at the output layer makes the network shift invariant (i.e. insensitive to the exact positioning of the gesture). Figure 3.12 shows our TDNN architecture for the experiments, where positive values are shown as gray squares and negative values as black squares. The inputs to our TDNN are vectors of (x, y, v, θ) for motion trajectories extracted from a gesture image sequence, where x, y are positions with respect to the center of the head, and v, θ are magnitudes and angle of velocity respectively; the outputs are the gesture classes; and the learning mechanism is error backpropagation.

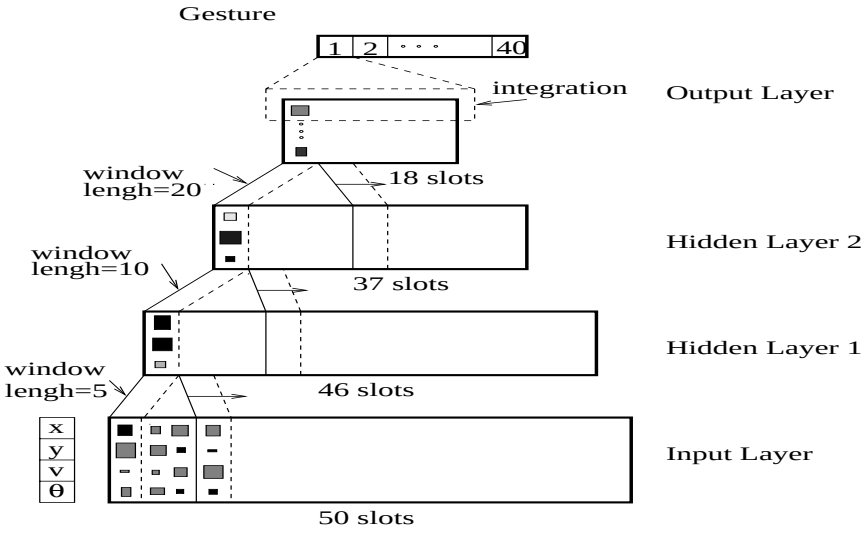


Figure 3.12: Architecture of TDNN

3.7 Experiments

We use a video database of 40 ASL signs for experiments. Each video consists of an ASL sign which lasts about 3 to 5 seconds at 30 frames per second with image size of 160×120 in Quicktime format. Figure 3.1 shows one complex ASL gesture from the sequence "cheerleader." Note that the hand movement consists of rotation and repetitions. Each image sequence of the 40 gestures in the experiment has 80 to 120 frames. Discarding the frames in which palms do not appear in the images (i.e. frames in starting and ending phase), each image sequence has about 50 frames. Motion regions with skin color are identified by their chromatic characteristics. These regions are then merged into palm and head regions shown in Figure 3.3 based on geometric analysis discussed in Section 3.4. Affine parameters of matched palm regions are computed, which give pixel motion trajectories for each image pair. By concatenating the trajectories for consecutive image pairs, continuous motion trajectories are generated. Figures 3.4 shows the extracted motion trajectories from a number of frames and Figure 3.9 shows the trajectories from the whole image sequence. Note that the motion trajectories of palm region match the movement in the real scene well.

Training of TDNN is performed on the corpus of 80% of the extracted dense (38 on the average) trajectories from each gesture, using an error backpropagation algorithm. The rest 20% of the trajectories are then used for testing. Based on the experiments with 40 ASL gestures, the average recognition rate on the training trajectories is 98.14% and the average recognition rate on the unseen test trajectories is 93.42%. Since dense motion trajectories are extracted from each image sequence, the recognition rate for each gesture can be improved by a "voting" scheme (i.e. the majority rules) on the classification result of each individual trajectory. The resulting average recognition rate on the training and testing sets for gesture recognition are 99.02% and 96.21%, respectively.

3.8 Discussion and Conclusion

We have described an algorithm to extract and recognize motion patterns using trajectories. For concreteness, the experiments have been carried out to recognize hand gestures in ASL. Motion segmentation is performed to generate regions with uniform motion. Moving regions with salient

features are then extracted using color and geometric information. The affine transformations associated with these regions are then concatenated to generate continuous trajectories. These motion trajectories encode the dynamic characteristics of hand gestures and are classified by a time-delay neural network. Our experiments demonstrate that hand gestures can be recognized, with high accuracy, using motion trajectories.

The contributions of this work can be summarized as follows. First, a general method that extracts motion trajectories is developed. This is in contrast to much work on gesture recognition that uses color histogram tracker [179] [37] [12], magnetic sensors [209], hand drawn template [79], and stereo [205] to obtain a representation of the gesture. Second, we use a TDNN to recognize gestures based on the extracted trajectories. Using an ensemble of trajectories helps achieve high recognition rates. It would be interesting to compare these recognition rates with those obtained using other recognition methods such as HMM, CONDENSATION algorithm [79] [12] and principal curve [209].

Chapter 4

Skin Color Model

Human skin color has been used and proved to be an effective feature in many applications from human face detection to hand tracking. However, most studies use either simple thresholding or a single Gaussian distribution to characterize the properties of skin color. Although skin colors of different races fall into a small cluster in normalized RGB or HSV color space, we find that a single Gaussian distribution is neither sufficient to model human skin color nor effective in general applications. Further, previous approaches use small collections of images to estimate the density function but do not validate the models by verifying the statistical fit of the chosen model to the data. The work in this chapter is aimed at estimating the properties of human skin color using the Michigan face database (http://www.engin.umich.edu/faces/) which consists of 2,447 images of human faces from different ethnic groups. More than 9.5 million skin color pixels are used to build a skin color model.

Within the framework of the Gaussian mixture model, two fundamental questions can be asked about the validity of the model. First, on the question of normality and homoscedasticity of the specified components, a test proposed by Hawkins [71] can be used to test both properties simultaneously. The second question is about of the number of Gaussian components used in the mixture. A test for the smallest number of components compatible with the data can be formulated in terms of the likelihood ratio criterion, although unfortunately it does not have its usual asymptotic distribution under the null hypothesis. Basford and McLachlan proposed the use of several measures using the bootstrap method [127] to quantify the strength of the obtained mixture model. In this work, these two tests are carried out to formally support the estimated model.

This chapter is organized as follows. We give a brief description of Gaussian mixture model and parameter estimation in Section 4.1. Two statistical tests on the normality and the number of are reviewed in Section 4.2. Section 4.3 provides the experimental results and applications on the obtained model are presented in Section 4.4. We conclude this chapter with comments in 4.5.

4.1 Proposed Mixture Model

In this section, we give a brief description of a finite Gaussian mixture model and then use it to model the distribution of skin color pixels.

4.1.1 Gaussian Mixture Model

Under the finite mixture models to be fitted in this work, each skin color pixel $\mathbf{x}$ can be viewed as arising from a super population G which is a mixture of a finite number, g, of populations $G_1, \ldots, G_g$ in some proportions $\pi_1, \ldots, \pi_g$, respectively, where

$$\sum_{i=1}^{g} \pi_i = 1 \text{ and } \pi_i \geq 0$$

The probability density function (p.d.f) of an observation $\mathbf{x}$ (of dimensionality d) in the finite mixture form is

$$p(\mathbf{x};\phi) = \sum_{i=1}^{g} \pi_i \cdot p_i(\mathbf{x};\theta) = \sum_{i=1}^{g} \pi_i \cdot p(\mathbf{x}|i;\theta) = \sum_{i=1}^{g} \pi_i \cdot \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{\frac{1}{2}}} \exp^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T(\Sigma_i)^{-1}(\mathbf{x}-\mu_i)}$$

where π_i is the mixing parameter, $p_i(\mathbf{x};\theta)$ is the p.d.f. corresponding to G_i, and θ denotes the vector of all unknown parameters associated with the parametric forms adopted for these g component densities. For the case of multivariate Gaussian components, θ consists of the elements of the mean vectors μ_i, and the distinct elements of the covariance matrices Σ_i for $i = 1, \ldots, g$. The vector

$$\phi = (\pi^T, \theta^T)^T$$

of all unknown parameters belongs to some parameter space Ω and is estimated using the EM algorithm [154] in Section 4.1.2.

4.1.2 Estimating Parameters Using EM Algorithm

Various procedures have been developed to determine the parameters of a Gaussian mixture model from a set of data. Here we briefly describe the EM algorithm for parameter estimation. A review on parameter estimation using maximum likelihood techniques can be found in a paper by Render and Walker [154].

For the case of Gaussian components, the mixture density contains the following adjustable parameters: π_i, μ_i and Σ_i (where $i = 1, \ldots, g$). The negative log-likelihood for the data set is given by

$$E = -\ln \mathcal{L} = -\sum_{j=1}^{n} \ln p(\mathbf{x}_j) = -\sum_{j=1}^{n} (\sum_{i=1}^{g} \pi_i p(\mathbf{x}_j|i)) \tag{4.1}$$

which can be regarded as an error function. Maximizing the likelihood $\mathcal{L}$ is then equivalent to minimizing E.

The EM algorithm begins by making some initial guess for the parameters of the Gaussian mixture model, which we call the "old" parameter values. We can then evaluate the new parameters using the following equations. This will give a revise estimate for the parameters, which we call the "new" parameter values, for which we might hope the value of error function is smaller. These parameter values then become the "old" values, and the process is repeated.

We write the change in error function when we replace the old parameter values by the new values in the form

$$\Delta^{t+1} = E^{t+1} - E^t = -\sum_{j}^{n} \ln(\frac{p^{t+1}(\mathbf{x}_j)}{p^t(\mathbf{x}_j)}) \tag{4.2}$$

where $p^{t+1}(\mathbf{x})$ denotes the probability density evaluated using the new values for the parameters, while $p^t(\mathbf{x})$ represents the density evaluated using the old parameter values.

By setting the derivatives of Δ^{t+1} to zero (see Render and Walker's review [154] for details), we obtain the following update equations for the parameters of the mixture model:

$$\mu_i^{t+1} = \frac{\sum_{j=1}^{n} p^t(i|\mathbf{x}_j)\mathbf{x}_j}{\sum_{j=1}^{n} p^t(i|\mathbf{x}_j)} \tag{4.3}$$

$$\Sigma_i^{t+1} = \frac{\sum_{j=1}^{n} p^t(i|\mathbf{x}_j)||\mathbf{x}_j - \mu_i^t||^2}{\sum_{j=1}^{n} p^t(i|\mathbf{x}_j)} \tag{4.4}$$

$$\pi_i^{t+1} = \frac{1}{n}\sum_{j=1}^{n} p^t(i|\mathbf{x}_j) \tag{4.5}$$

where

$$p^t(i|\mathbf{x}_j) = \frac{p^t(\mathbf{x}_j|i)\pi^t(i)}{p^t(\mathbf{x}_j)} \tag{4.6}$$

4.2 Statistical Tests

In order to determine the goodness-of-fit of a Gaussian mixture, we use Hawkins' method to test the normality and homoscedasticity of the mixture models and the bootstrap method to test the number of components required in the model. These methods are briefly described in the following sections. For details, see the treatments in Titterington [196] and McLachlan [127].

4.2.1 Hawkins' Test for Normality and Homoscedasticity

Under the normality assumption with equal covariance matrices (homoscedasticity) for the component distribution in a Gaussian mixture, we have a null hypothesis

$$H_0 : \mathbf{x} \sim N(\mu_i, \Sigma) \text{ in } G_i \ (i = 1, \ldots, g) \tag{4.7}$$

on the basis of the classified data $\mathbf{y}_{ij}$ $(1 = 1, \ldots, g; j = 1, \ldots, m_i)$. Let

$$\bar{\mathbf{y}}_i = \sum_{j=1}^{m_i} \mathbf{y}_{ij}/m_i \tag{4.8}$$

and

$$\mathbf{S}_i = \sum_{j=1}^{m_i} (\mathbf{y}_{ij} - \bar{\mathbf{y}}_i)(\mathbf{y}_{ij} - \bar{\mathbf{y}}_i)^T/(m_i - 1) \tag{4.9}$$

for $i = 1, \ldots, , g$ and let

$$\mathbf{S} = \sum_{i=1}^{g} (m_i - 1)\mathbf{S}_i/(m - g) \text{ where } m = \textstyle\sum_{i=1}^{g} m_i \tag{4.10}$$

The Mahalanobis squared distance between $\mathbf{y}_{ij}$ and $\bar{\mathbf{y}}_i$ with respect to $\mathbf{S}$ is denoted by $D(\mathbf{y}_{ij}, \bar{\mathbf{y}}_i; \mathbf{S})$, so that

$$D(\mathbf{y}_{ij}, \bar{\mathbf{y}}_i; \mathbf{S}) = (\mathbf{y}_{ij} - \bar{\mathbf{y}}_i)^T \mathbf{S}^{-1} (\mathbf{y}_{ij} - \bar{\mathbf{y}}_i)$$

The notation for the Mahalanobis squared distance between two vectors with respect to some positive definite symmetric matrix is to be used in this work. For a given population G_i, the test considers the Mahalanobis squared distance between each $\mathbf{y}_{ij} (j = 1, \ldots, m_i)$ and the mean of the sample from G_i, but where each $\mathbf{y}_{ij}$ is deleted from the sample if it severely contaminates the estimates of the mean and covariance matrix of $G_i (i = 1, \ldots, g)$. Accordingly, the Mahalanobis squared distance

$$D(\mathbf{y}_{ij}, \bar{\mathbf{y}}_i; \mathbf{S}_{(ij)}) \tag{4.11}$$

is computed where $\bar{y}_{i(ij)}$ and $\mathbf{S}_{(ij)}$ denote the resulting values of $\bar{\mathbf{y}}_i$ and $\mathbf{S}$ after the deletion of $\mathbf{y}_{ij}$ from the data. It follows that under H_0,

$$c(m_i, \nu) D(\mathbf{y}_{ij}, \bar{y}_{(ij)}; \mathbf{S}_{(ij)}) \tag{4.12}$$

is distributed according to an F distribution with p and $\nu = m - g - p$ degrees of freedom, where

$$c(m_i, \nu) = ((m_i - 1)\nu)/((m_i p)(\nu + p - 1))$$

To avoid the recomputation of $\bar{\mathbf{y}}_i$ and $\mathbf{S}$ after the deletion of each $\mathbf{y}_{ij}$ from the data, it was shown that (4.12) can be computed using the result that

$$c(m_i, \nu) D(\mathbf{y}_{ij}, \bar{y}_{i(ij)}; \mathbf{S}_{(ij)}) = \frac{(\nu m_i / p) D(\mathbf{y}_{ij}, \bar{\mathbf{y}}_i; \mathbf{S})}{(\nu + p)(m_i - 1) - m_i D(\mathbf{y}_{ij}, \bar{\mathbf{y}}_i; \mathbf{S})} \tag{4.13}$$

Let a_{ij} denote the area to the right of the observed value of (4.13) under the $F_{p,\nu}$ distribution, then under H_0 we have that

$$H_{0i} : a_{i1}, \ldots, a_{im_i} \overset{iid}{\sim} U(0,1) \ (i = 1, \ldots, g) \tag{4.14}$$

holds approximately, where $U(0,1)$ denotes the uniform distribution on the unit interval. The result of (4.14) is only an approximate as for a given i, the a_{ij} are only independent exactly as $m_i \to \infty$, due to the presence of the estimates of μ_i and Σ in the formation of (4.11). Hawkins [71] has reported empirical evidence which suggests that subsequent steps in his test which treat (4.14) as if it were an exact result should be approximately valid.

A close inspection of the tail areas a_{ij} including $Q - Q$ plots can be used to detect departures from the g hypotheses H_{0i}, and hence from the original hypothesis H_0. In conjunction with this detailed analysis, Hawkins advocated the use of Anderson-Darling statistic for assessing (4.14), as this statistic is particularly sensitive to fit in the tails of the distribution. The Anderson-Darling statistic can be computed for the sample of m_i values $a_{ij} (j = 1, \ldots, m_i)$ by

$$W_i = -m_i - \sum_{j=1}^{m_i} (2j - 1)(\log a_{i(j)} + \log(1 - a_{i(m_i - j + 1)})/m_i \ (i = 1, \ldots, g)$$

where for each i, $a_{i(1)} \leq a_{i(2)} \leq \cdots \leq a_{i(m_i)}$ denote the m_i order statistics of the a_{ij}. In the

asymptotic resolution of each W_i into standard normal variances W_{ik} according to

$$W_i = \sum_{k=1}^{\infty} W_{ik}^2/(k(k+1)) \ (i = 1, \ldots, g)$$

where attention is focused on the first two components

$$W_{i1} = -(3/m_i)^{1/2} \sum_{j=1}^{m_i} (2a_{i(j)} - 1)$$

and

$$W_{i2} = -(5/m_i)^{1/2} \sum_{j=1}^{m_i} \frac{1}{2}(3(2a_{i(j)} - 1)^2 - 1)$$

Similarly, the Anderson-Darling statistic W_T and its first two components W_{T1} and W_{T2} can be computed for the single sample where all the a_{ij} are combined.

Some simulations performed by Hawkins [71] suggest that the size of the test will be approximately 0.1 if (4.14) is rejected if any W_i exceeds 2.5 (the asymptotic 95th percentile) or any W_{ik} (k=1,2) exceeds 2.54 in magnitude.

4.2.2 Statistical Test for the Number of Components

The "bootstrap" method, first introduced by Efron [46], is a powerful technique which permits the variability in a random quantity to be assessed using just the data at hand. An estimate $\hat{F}$ of the underlying distribution is formed from the observed sample. Conditional on the later, the sampling distribution of the random quantity of interest with F replaced by $\hat{F}$, defines its so-called bootstrap distribution, which provides an approximation to its true distribution. It is assumed that $\hat{F}$ has been so formed that the stochastic structure of the model has been preserved. Usually, it is impossible to express the bootstrap distribution in simple form, and it must be approximated by Monte Carlo methods whereby pseudo-random samples (bootstrap samples) are drawn from $\hat{F}$. The bootstrap method can be implemented nonparametrically by using the empirical distribution function constructed from the original data. In the following application the bootstrap is applied in a parametric framework in which the bootstrap samples are drawn from the parametric likelihood

estimate of the underlying distribution function.

The log likelihood ratio statistic for the test of the null hypothesis $H_0 : g = g_1$ groups versus the alternative $H_1 : g = g_2$ can be bootstrapped as follows. Proceeding under H_0, a bootstrap sample is generated from a mixture of g_1 groups where, in the specified from their densities, unknown parameters are replaced by their likelihood estimates formed under H_0 from the original sample. The value of $-2\log\lambda$ is computed fro the bootstrap sample after fitting mixture models for $g = g_1$ and $g = g_2$ in turn to it. This process is repeated independently a number of times K, and the replicated values of $-2\log\lambda$ formed from the successive bootstrap samples provide an assessment of the bootstrap, and hence of the true, null distribution of $-2\log\lambda$. It enables an approximation to be made to the achieved level of significance P corresponding to the value of $-2\log\lambda$ evaluated from the original sample.

If a very accurate estimate of the P-value were required, then K may have to be very large. Indeed for less complicated models than mixtures, Efron et al. [46] have shown that whereas 50 to 100 bootstrap replications may be sufficient fro standard error and bias estimation, a larger number, say 350, are needed to give a useful estimate of a percentile or P-value, and many more for a highly accurate assessment. Usually, however, there is no interest in estimating a P-value with high precision. Even with a limited replication number K, the amount of computation involved is still considerable, in particular for values of g_1 and g_2 not close to one.

In the narrower sense where the decision to be made concerns solely the rejection or retention of the null hypothesis at a specified significance level α, Aitkin [3] noted how analogous to the Monte Carlo test produce of Hope [73], the bootstrap replications can be used to provide a test of approximate size α. The test which rejects H_0 if $-2\log\lambda$ for the original data is greater than the hth smallest of its K bootstrap replications, has size

$$\alpha = 1 - j/(K+1) \tag{4.15}$$

approximately. For if any difference between the bootstrap and true null distribution of $-2\log\lambda$ is ignored, then the original and subsequent bootstrap values of $-2\log\lambda$ can be treated as the realizations of a random sample of size $K+1$, and the probability that a specified member is greater than j of the others is $1 - j/(K+1)$.

The result of (4.15) applies to the unconditional size of the test and not to its size conditional on the K bootstrap values of $-2\log\lambda$. For a specified significance level α, the values of j and K can be appropriately chosen according to (4.15). For example, for $\alpha = 0.05$, the smallest value of K needed is 19 with $j = 19$. As cautioned above on the estimation of the P-value for the likelihood ratio test, K needs to be very large to ensure an accurate assessment. For the 0.05 level test of a single normal population versus a mixture of two normal homoscedastic populations, McLachlan [126] performed some simulations to demonstrate the improvement in the power as K increased from 19 through 39 to 99.

4.3 Experimental Results

Each image from the Michigan face database is segmented using the multiscale transform [2] and the skin color regions are selected. The total number of analyzed pixels is 9,565,862 (skin color pixels) where each sample consists of three values (r,g,b). To reduce the dependence on the lighting condition, each sample is transformed from RGB to CIE LUV color space and then the lightness value is discarded. Figure 4.1 shows the resulting 2D histogram (downsampled by a factor of 10) of skin color histogram of $\mathbf{x}$ ($\mathbf{x} = (u, v)^T$). It is clear that a single Gaussian density function is not sufficient to model the distribution of skin color.

4.3.1 Estimated Density Function

We use the EM algorithm to estimate the parameters of the Gaussian mixture. The samples are initially labeled using the k-means clustering where k is equal to g (the number of components in the mixture model). In our experiments, k is set to 2 because the histogram in Figure 4.1 can be modeled as a mixture model with 2 components (i.e. $g = 2$). The parameters, ϕ, are estimated using the E-step (expectation) and M-step (maximization) iteratively [154]. The estimated density function (Figure 4.2) perceptually fits the histogram of the samples (Figure 4.1(a)). It is evident that a finite Gaussian mixture model is more appropriate for estimating the density function of human skin color. To further support the argument, statistical tests on the normality and the number of components are performed.

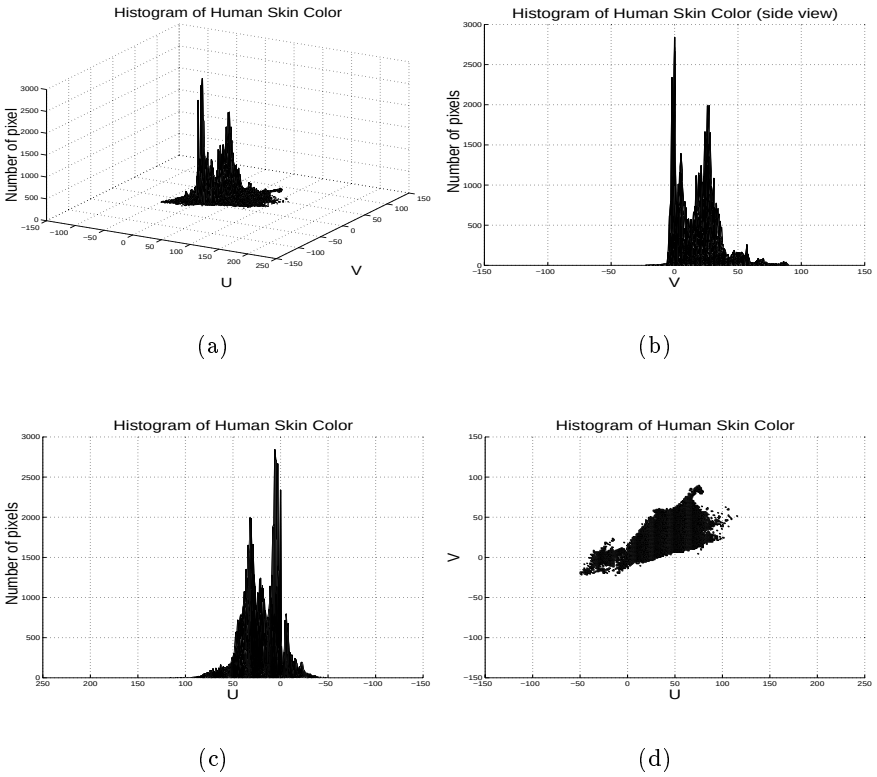


Figure 4.1: Histogram of skin color (downsampled by a factor of 10) viewed from different angles

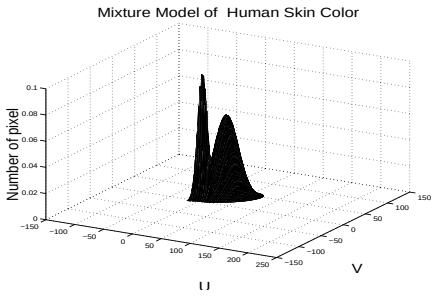


Figure 4.2: Estimated Density Function

4.3.2 Hawkins' Test on Normality and Homoscedasticity

The result of Hawkins' test of

$$H_0 : \mathbf{x} \sim \mathcal{N}(\mu_i, \Sigma) \text{ in } G_i \ (i = 1, 2)$$

for normality and homoscedasticity are given in Table 1. They are for the application of Hawkins' test to the data in their known classified form. That is, for each observation $\mathbf{x}_{ij} (i = 1, 2; j = 1, \ldots, n_i)$, the tail area a_{ij} to the right of the Anderson-Darling statistic [71] was computed under the F distribution. The Anderson-Darling statistic and its first two asymptotic $\mathcal{N}(0, 1)$ components for the a_{ij} from G_i and for the totality of the a_{ij} are useful in interpreting qualitatively departures from H_0. From Table 1, the difference in sign and significance of the first components of the Anderson-Darling statistics for the individual populations indicate hetroscedasticity while the nonsignificance of the Anderson-Darling statistic and its components for the totality of the a_{ij} gives a fair indication that G_1 and G_2 are bivariate normal.

Table 4.1: Results of Hawkins' test for normality and homoscedasticity(applied to data in known classified form)

Source	Anderson-Darling statistic	Components of Anderson-Darling statistic	
		First	Second
G1	3.07	-2.14	-1.50
G2	2.30	1.90	0.97
Total	0.15	0.12	-0.31

4.3.3 Bootstrap Test on the Number of Components

Now that we know the normality and hetroscedasticity of the data set, it is of interest to see if the likelihood ratio test of

$$H_0 : g = 2 \text{ versus } H_1 : g = 3$$

would lead to the rejection of the null hypothesis of two components.

For the bootstrap test suggested by McLachlan [127], H_0 is rejected at a nominal α level if

$-2\log\lambda$ evaluated for the original sample exceeds the $(1-\alpha)(K+1)$th smallest of K bootstrap values subsequently replicated for this statistic, where $\alpha(K+1)$ is an integer. Given the amount of computation involved, K was limited to 19. Proceeding under H_0, 19 bootstrap samples were generated from two component Gaussian mixture. For each of the 19 bootstrap samples a mixture of $g = 3$ with unequal covariance matrices was fitted, and the increase in the log likelihood over that for a single population calculated. As the value of 10.6 for $-2\log\lambda$ from the original samples was less than 6 of 19 replications, H_0 would clearly not be rejected in favor of H_1. The conclusion, based on $K = 19$, is that two component Gaussian mixture hypothesis is not rejected.

4.4 Applications

One application of using the estimated mixture density is to detect human faces in an image database. Given an image, a multiscale segmentation is performed to obtain homogeneous regions. Each pixel is classified to be skin color if its probability measure is above a threshold. Each region is then recognized as a skin area if the most pixels of the regions have high probability to be skin color. Figure 4.3(a) shows an image and Figures 4.3(a), 4.3(b) shows the results after skin detection using simple thresholds. The result using estimated mixture density function is shown in Figure 4.3(c). It is clear that a mixture model has better performance than a single Gaussian distribution function in detecting skin regions.

(a) original image

(b) extracted skin regions by a Gaussian density function with low threshold

(c) extracted skin regions by a Gaussian density function with high threshold

(d) extracted skin regions by a mixture density function

Figure 4.3: Original image and results of skin detection

Skin color alone is usually not sufficient in detecting human faces or hands. For example,

people wear clothes whose colors are similar to skin color as shown in Figure 4.3. Nevertheless, a good estimated density function of skin color is very useful and effective in simplifying the tasks of skin area detection. Using the skin color and structure information, human faces can be detected robustly [216].

In our experiments, if more than 70% of the pixels in a region are classified to be skin color, then the region is recognized as a skin area. Figure 4.4 shows an image sequence and Figure 4.5 shows the results of skin detection.

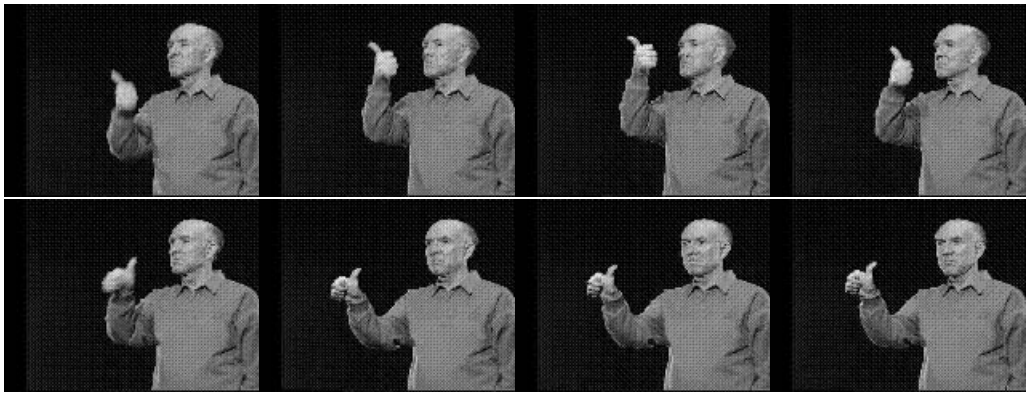


Figure 4.4: Image sequence of ASL sign "any" (time increases left to right and top to bottom)

Figure 4.5: Detected skin areas in the sequence in Figure 4.4

4.5 Discussion and Conclusion

In this chapter, a Gaussian mixture model for human skin color is introduced. It is evident, both from the histogram of large samples and the estimated density function, that a Gaussian mixture

is more appropriate than a single Gaussian function in estimating the distribution of skin color. Statistical tests on the normality and the number of components are performed to justify the hypotheses. Applications using the estimated density function show that the estimated density function of skin color is useful and effective in detecting skin regions.

Chapter 5

Face Detection

In this chapter, we present three algorithms to detect faces in color and gray-level images. The first method utilizes structure and color information to detect faces in color images [216]. The other two methods use density functions to model the distributions of human face in the mixture of subspaces [222] [223].

5.1 Introduction

Images of human faces are central to intelligent human computer interaction. Much research is being done involving face images, including face recognition, face tracking, pose estimation, expression recognition and gesture recognition. However, most existing methods on these topics assume human faces in an image or an image sequence have been identified and localized. To build a fully automated system that extracts information from images of human faces, it is essential to develop robust and efficient algorithms to detect human faces. Given a single image or a sequence of images, the goal of face detection is to identify and locate all of the human faces regardless of their positions, scales, orientations, poses and lighting conditions. This is a challenging problem because human faces are highly non-rigid objects with a high degree of variability in size, shape, color and texture. Most recent methods for face detection can only detect upright, frontal faces under certain lighting conditions. In this chapter, we present on method based on structure and color to detect faces in color images. Next, we present two face detection methods that use mixtures of linear subspaces to detect faces with different features and expressions, in different poses, and under different lighting conditions.

In the first method, a human skin color model is built to capture the chromatic properties based on multivariate statistical analysis. Given a color image, multiscale segmentation is used to generate homogeneous regions at multiple different scales. From the coarsest to the finest scale, regions of skin color are merged until the shape is approximately elliptic. Postprocessing is performed to determine whether a merged region contains a human face and include the facial features of non-skin color such as eyes and mouth if necessary. Experimental results show that human faces in color images can be detected regardless of size, orientation and viewpoint.

Since the images of a human face lie in a complex subset of the image space that is unlikely to be modeled by a single linear subspace, we use a mixture of linear subspaces to model the distribution of face and nonface patterns. The second detection method is an extension of factor analysis. Factor analysis (FA), a statistical method for modeling the covariance structure of high dimensional data using a small number of latent variables, has some analogies with principal component analysis (PCA). However PCA, unlike FA, does not define a proper density model for the data since the cost of coding a data point is equal anywhere along the principal component subspace (i.e. the density is unnormalized along these directions). Further, PCA is not robust to independent noise in the features of the data since the principal components maximizes the variances of the input data, thereby retaining unwanted variations. Hinton et al. have applied FA to digit recognition and they compare the performance of PCA and FA models [72]. A mixture model of factor analyzers has recently been extended [58] and applied to face recognition [54]. Both studies show that FA performs better than PCA in digit and face recognition. Since pose, orientation, expression, and lighting affect the appearance of a human face, the distribution of faces in the image space can be better represented by a mixture of subspaces where each subspace captures certain characteristics of certain face appearances. We present a probabilistic method that uses a mixture of factor analyzers (MFA) to detect faces with wide variations. The parameters in the mixture model are estimated using an EM algorithm.

The third method that we present uses Fisher Linear Discriminant (FLD) to project samples from a high dimensional image space to a lower dimensional feature space. Recently, the Fisherface method has been shown to outperform the widely used Eigenface method in face recognition [10]. The reason for this is that FLD provides a better projection than PCA. In the second proposed

method, we decompose the training face and nonface samples into several classes using Kohonen's Self Organizing Map (SOM). From these labeled classes, the within-class and between-class scatter matrices are computed, thereby generating the optimal projection based on FLD. For each subspace, we use a Gaussian to model each class-conditional density function where the parameters are estimated based on maximum likelihood [41]. To detect faces, each input image is scanned with a rectangular window in which the class-dependent probability is computed. The maximum likelihood decision rule is used to determine whether a face is detected or not.

To capture the variations in face patterns, we use a set of 1,681 face images from Olivetti [171], UMIST [65], Harvard [66], Yale [10] and FERET [141] databases. Both methods have been tested using the databases in [161] [188] to compare their performances with other methods. Our experimental results on the data sets used in [161] [188] (which consist of 225 images with 619 faces) show that our methods perform as well as the reported methods in the literature, yet with fewer false detects. To further test our methods, we collect a set of 80 images containing 252 faces. This data set is rather challenging since it contains profile faces, faces with expressions and faces with heavy shadows. Our methods are able to detect most of these faces regardless of their poses, facial expressions and lighting conditions. Furthermore, our methods have fewer false detects than other methods.

5.2 Detecting Faces in Color Images

5.2.1 Multiscale Segmentation

Multiscale image segmentation is used to extract a hierarchy of regions for matching. Segmentation is achieved using a transform function, which maps the feature primitives to a family of attraction force fields, defined by

$$\mathbf{F}(x,y;\sigma_g(x,y),\sigma_s(x,y)) = \int\int_R d_g(\Delta I,\sigma_g(x,y))\cdot d_s(\vec{r},\sigma_s(x,y))\frac{\vec{r}}{||\vec{r}||}dwdv \tag{5.1}$$

where $R = domain(I(u,v))\backslash\{(x,y)\}$ and $\vec{r} = (v-x)\vec{i} + (w-y)\vec{j}$. The details of this transform and the segmentation method can be found in [2]. Here we review its basic characteristics. The parameter σ_g is a homogeneity scale which reflects the homogeneity of the region into which pixel

groups and σ_s is a spatial scale that controls the neighborhood from which the force on the pixel is computed. The force field encodes the region structure in a manner which allows easy extraction.

The homogeneity between two pixels is given by the Euclidean distance between the associated m-dimensional feature vectors (e.g., $m = 3$ for a color image)

$$\Delta I = |I(x, y) - I(v, w)| \tag{5.2}$$

The spatial scale parameter, σ_s, controls the spatial distance function, $d_s(\cdot)$, and the homogeneity scale parameter, σ_g, controls the homogeneity distance function, $d_g(\cdot)$. One possible form for these functions satisfying criteria discussed in [2] are unnormalized Gaussian

$$\begin{aligned} d_g(\Delta I, \sigma_g) &\sim \sqrt{2\pi\sigma_g^2} N_{\Delta_I}(0, \sigma_g^2) \\ d_s(\vec{r}, \sigma_s) &\sim \begin{cases} \sqrt{2\pi\sigma_s^2} N_{||\vec{r}||}(0, \sigma_s^2), & ||\vec{r}|| \leq 2\sigma_s \\ 0, & ||\vec{r}|| > 2\sigma_s \end{cases} \end{aligned} \tag{5.3}$$

With the above definition of force field $\mathbf{F}$, pixels are grouped together into regions whose boundaries correspond to diverging force vectors in $\mathbf{F}$ and whose skeletons correspond to converging force vectors in $\mathbf{F}$. An increase in σ_g causes less homogeneous structures to be encoded and an increase in σ_s causes large structures to be encoded.

Figure 5.1 (a) shows an original image. The following three images show the segmented image with all pixels in regions at three different scales ($\sigma_g = 40, 20, 6$ for (b), (c), and (d)) replaced by a unique gray value for each region. The identified regions are visually correct and the region boundaries align with the actual boundaries at all scales. Note that all segmentation parameters of the transform are selected automatically, eliminating the need to make *a priori* assumptions about either the geometric or homogeneity characteristics of the structure.

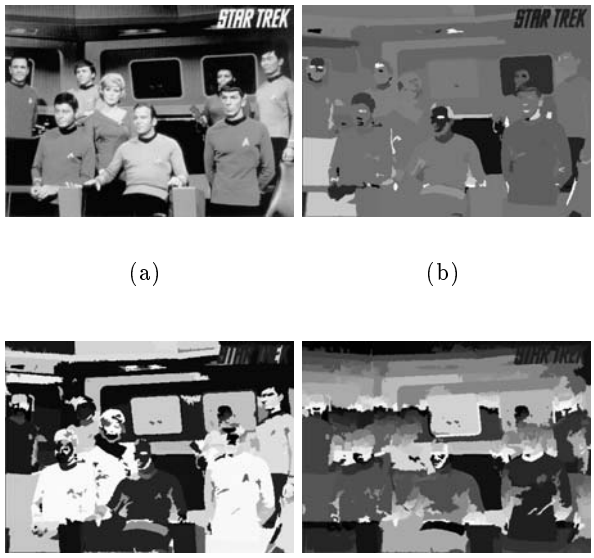


Figure 5.1: Results of multiscale segmentation

5.2.2 Human Skin Color Model

Although different people have different colors in appearance, several studies have showed the major difference lies in intensity rather than color itself [63] [214]. To build a skin color model, we use CIE LUV color space and discard the luminance value. We use the Gaussian mixture skin color model described in Chapter 4. The parameters of the distribution are estimated using an EM algorithm. A pixel is identified to have skin color if its corresponding probability is greater than a threshold (0.5) and a region is classified to have human skin color if most (above 70%) of its pixels have skin color. Figure 5.2(a) shows the segmented regions of Figure 5.1(a) that have skin color. Note that skin color helps in detecting human face, but skin color alone cannot detect human faces correctly.

5.2.3 Geometric Analysis and Postprocessing

Since the shapes of human faces are approximately elliptic, skin color regions are merged until the shape of the merged region is approximately elliptic. The orientation of an ellipse can be calculated by the moments of inertia [74]. The extents of the major and minor axes of the ellipse are approximated by the extents of the region along the axis directions and the degree of fit of the

(a) (b)

Figure 5.2: Intermediate and final results of face detection

ellipse is determined by the number of pixels that fall into that shape specified by the computed parameters. A merged skin color region is a human face candidate if the ratio of major axis to minor axis is less than a threshold (1.7). If a candidate region has some darken regions or some holes inside the merged region, it is classified as a human face. The rationale is that a face contains eyes or mouth and these facial features have darker colors than skin. If these facial features are included in the skin color region, the color of the facial features should be relatively darker than the rest of the region. If the facial features are not classified as skin color regions, some empty areas should exist inside the candidate region. In this case, the segmented regions of the empty areas are included as part of a face. Figure 5.2(b) shows the final result of detected human faces in Figure 5.2.

5.2.4 Experimental Results

In this section, we present experimental results (color images of the experiments can be found at http://vision.ai.uiuc.edu/mhyang/facedetection.html) on color images collected from the Internet. Each detected human faces is shown with an enclosing ellipse. Figure 5.3(a) shows that human faces of different viewpoints and sizes are detected. Note that faces from different ethnic backgrounds are detected. Faces of different orientations can also be identified as shown in Figure 5.3(b) Figure 5.4 gives an example that human faces with glasses and sideburns can still be detected. Finally, Figure 5.5 demonstrates that human faces can be extracted in a color image with complex background.

(a) (b)

Figure 5.3: Faces of different size and orientation

Figure 5.4: Faces with different features

Figure 5.5: Human faces in complex background

5.2.5 Conclusion

In this section, we have presented a new method to detect human faces in color images. The proposed method utilizes a multiscale transform to segment images into homogeneous regions and extracts skin regions based on a skin color model. These regions are then merged until the shape is approximately elliptic and verified. Experimental results show that the proposed method can detect human faces in color image regardless of size, orientation and viewpoint.

5.3 Mixture of Factor Analyzers

In the first subspace method, we fit the mixture model of factor analyzers to the training samples using an EM algorithm and obtain a distribution of face patterns. To detect faces, each input image is scanned with a rectangular window in which the probability of the current input being a face pattern is calculated. A face is detected if the probability is above a predefined threshold. We briefly describe factor analysis and a mixture of factor analyzers in this section. The details of these models can be found in [6] [58].

5.3.1 Factor Analysis

Factor analysis is a statistical model in which the observed vector is partitioned into an unobserved systematic part and an unobserved error part. The systematic part is taken as a linear combination of a relatively small number of unobserved factor variables while the components of the error vector are considered as uncorrelated or independent. From another point of view, factor analysis gives a description of the interdependence of a set of variables in terms of the factors without regard to the observed variability. In this model, a d-dimensional real-valued observable data vector x is modeled using a p-dimensional vector of real-valued factors z where p is generally much smaller than d. The generative model is given by:

$$x = \Lambda z + u \tag{5.4}$$

where Λ is known as the *factor loading matrix*. The factors z are assumed to be $\mathcal{N}(0, I)$ distributed (zero-mean independent normals with unit variance). The d-dimensional random variable u is

distributed $\mathcal{N}(0, \Psi)$ where Ψ is a diagonal matrix, due to the assumption that the observed variables are independent given the factors. According to this model, x is therefore distributed with zero mean and covariance $\Sigma = \Lambda\Lambda^T + \Psi$. The goal of factor analysis is to find the Λ and Ψ that best model the covariance structure of x. The factor variables z model correlations between the elements of x, while the u variables account for independent noise in each element x. The p factors play the same role as the principal components in PCA, i.e. they are informative projections of the data. Given Λ and Ψ, the expected value of the factors can be computed through the linear projections:

$$E[z|x] = \beta x \tag{5.5}$$

$$E[zz^T|x] = I - \beta\Lambda + \beta x x^T \beta^T \tag{5.6}$$

where $\beta = \Lambda^T \Sigma^{-1}$.

5.3.2 Mixture Model

In this section, we consider a mixture of m factor analyzers (indexed by $f_j, j = 1, \ldots, m$) where each factor analyzer has the same number of p factors and each factor analyzer has a different mean μ_j. The generative model obeys the mixture distribution:

$$P(x) = \sum_{j=1}^{m} \int P(x|z, f_j) P(z|f_j) P(f_j) dz \tag{5.7}$$

where

$$P(z|f_j) = P(z) = \mathcal{N}(0, I) \tag{5.8}$$

$$P(x|z, f_j) = \mathcal{N}(\mu_j + \Lambda_j z, \Psi) \tag{5.9}$$

The parameters of this mixture model are $\{(\mu_j, \Lambda_j)_{j=1}^m, \pi, \Psi\}$ where π is the vector of adaptable mixing proportions, $\pi_j = P(f_j)$. The latent variables in this model are the factors z and the

mixture indicator variable f_j, where $f_j = 1$ when the data point is generated by the first factor analyzer.

Given a set of training images, the EM algorithm [39] is used to estimate $\{ (\mu_j, \Lambda_j)_{j=1}^m, \pi, \Psi\}$. For the E-step of the EM algorithm, we need to compute expectations of all the interactions of the hidden variables that appear in the log likelihood,

$$E[f_j z|x_i] = E[f_j|x_i]E[z|f_j, x_i] \tag{5.10}$$

$$E[f_j zz^T|x_i] = E[f_j|x_i]E[zz^T|f_j, x_i] \tag{5.11}$$

Defining

$$h_{ij} = E[f_j|x_i] \propto P(x_i, f_j) = \pi_j \mathcal{N}(x_i - \mu_j, \Lambda_j \Lambda_j^T + \Psi) \tag{5.12}$$

and using equations (5.5) and (5.9), we obtain

$$E[f_j z|x_i] = h_{ij}\beta_j(x_i - \mu_j) \tag{5.13}$$

where $\beta_j \equiv \Lambda_j^T(\Lambda_j\Lambda_j^T)^{-1}$. Similarly, using equations (5.6) and (5.11), we obtain

$$E[f_j zz^T|x_i] = h_{ij}(I - \beta_j\Lambda_j + \beta_j(x_i - \mu_j)(x_i - \mu_j)^T\beta_j^T) \tag{5.14}$$

The EM algorithm for mixture of factor analyzers can be stated as follows:

- **E-step**: Compute $E[f_j|x_i]$, $E[z\ |f_j,\ x_i]$ and $E[zz^T|\ f_j,\ x_i]$ for all data points i and mixture components j.
- **M-step**: Solve a set of linear equations for π_j, Λ_j, μ_j and Ψ.

The mixture of factor analyzers is essentially a reduced dimensionality mixture of Gaussians. Each factor analyzer fits a Gaussian to a portion of the data, weighted by the posterior probabilities, h_{ij}. Since the covariance matrix for each Gaussian is specified through the lower dimensional factor

loading matrices, the model has $mpd+d$, rather than $md(d+1)/2$ parameters dedicated to modeling covariance structure in high dimensions.

5.3.3 Detecting Face Patterns

To detect faces, each input image is scanned with a rectangular window in which the probability of there being a face pattern is estimated as given in equation (5.7). A face is detected if the probability is above a predefined threshold. In order to detect faces of different scales, each input image is repeatedly subsampled by a factor of 1.2 and scanned through for 10 iterations.

5.4 Mixture of Linear Spaces Using Fisher Linear Discriminant

In the second mixture model, we first use Kohonen's self-organizing map [101] to divide the face and nonface samples into 25 face classes and 25 nonface classes, thereby generating labels for the samples. Next, Fisher projection is computed based on all 50 classes to maximize the ratio of the between-class scatter (variance) and the within-class scatter (variance). The now labeled training set is projected from a high dimensional image space to a lower dimensional feature space, and a Gaussian distribution is used to model the class-conditional density function for each class where the parameters are estimated using the maximum likelihood principle. For detection, the conditional probability of each sample given each class is computed and the maximum likelihood principle is used to decide to which class the sample belongs. In our experiments, the reason that we choose 25 face and 25 nonface classes is because of the size of training set. If the number of classes is too small, the clustering results may be poor. On the other hand, we may not have enough samples to estimate the class-conditional density function well if we choose a large number of classes.

5.4.1 Labeling Samples Using Self-Organizing Map

In applying Fisher Linear Discriminant to find an projection, we need to know the class label of each training sample. However, such information is not available in the training samples. Therefore, we use Kohonen's Self-Organizing Map [101] to divide face samples into a finite number of classes. In our experiments, we divide the face sample images into 25 classes. After training, the final weight vector for each node is the centroid of the class, i.e. the prototype vector, which corresponds to

Figure 5.6: Prototype of each face class

the canonical face of each class. The same procedure is applied to nonface samples. Figure 5.6 shows the prototype of each class. It is clear that the sample face images with different poses and under different lighting conditions (intensity increases from the lower right corner to the upper left corner) have been classified into different classes.

5.4.2 Fisher Linear Discriminant

While PCA is commonly used to project face patterns from a high dimensional image space to a lower dimensional feature space, a drawback of this approach is that it defines a subspace such that it has the greatest variance of the projected sample vectors among all the subspaces. However, such projection is not suitable for classification since it may contain principal components which retain unwanted large variations. Therefore, the classes in the projected space may not be well clustered and instead smeared together [10] [54] [72]. Fisher Linear Discriminant is an example of a class specific method that finds the optimal projection for classification. Rather than finding a projection that maximizes the projected variance, FLD determines a projection, $y = W_{FLD}^T\ x$, that maximizes the ratio between the between-class scatter (variance) and the within-class scatter (variance). Consequently, classification is simplified in the projected space. Recently, it has been demonstrated that the Fisherface method outperforms the Eigenface method in face recognition

[10].

Consider a c-class problem, let the between-class scatter matrix be defined as

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T \tag{5.15}$$

and the within-class scatter matrix be defined as

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T \tag{5.16}$$

where μ_i is the mean of class X_i, and N_i is the number of samples in class X_i. The optimal projection W_{FLD} is chosen as the matrix with orthonormal columns which maximizes the ratio of the determinant of the between-class scatter matrix of the projected samples to the determinant of the within-class scatter matrix of the projected sampled, i.e.

$$W_{FLD} = \arg\max_{w} \frac{|W^T S_B W|}{|W^T S_W W|} = [w_1 \ w_2 \ \ldots \ w_m] \tag{5.17}$$

where $\{w_i | i = 1, 2, \ldots, m\}$ is the set of generalized eigenvectors of S_B and S_W, corresponding to the m largest generalized eigenvalues $\{\lambda_i | i = 1, 2, \ldots, m\}$. However, the rank of S_B is $c - 1$ or less because it is the sum of c matrices of rank one or less. Thus, the upper bound on m is $c - 1$ [41]. See [10] for details about a method to overcome singularity problems in computing W_{FLD}.

5.4.3 Class-Conditional Density Function

Once W_{FLD} is computed, the now labeled training set is projected to the $c-1$ dimensional feature space, i.e. $y = W_{FLD}^T \ x$, and a Gaussian distribution is used to model each class-conditional density (CCD) function, i.e. $P(y|X_i) = \mathcal{N}(\mu_{X_i}, \Sigma_{X_i})$ where $i = 1, \ldots, c$. The parameters, $\theta_{X_i} = (\mu_{X_i}, \Sigma_{X_i})\}$ of each CCD are the maximum likelihood estimates, i.e.

$$\hat{\mu}_{X_i} = \frac{1}{|X_i|} \sum_{y_k \in X_i} y_k \tag{5.18}$$

and

$$\hat{\Sigma}_{X_i} = \frac{1}{|X_i|} \sum_{y_k \in X_i} (y_k - \hat{\mu}_{X_i})(y_k - \hat{\mu}_{X_i})^T \tag{5.19}$$

5.4.4 Detecting Face Patterns

Each input image is scanned with a rectangular window to determine whether a face exists in the window or not. The decision rule for deciding whether an input window contains a face or not is based on maximum likelihood,

$$X^* = \arg\max_{X_i} P(y|X_i) \tag{5.20}$$

To detect faces of different scales, each input image is repeatedly subsampled by a factor of 1.2 and scanned through for 10 iterations.

5.5 Experiments

For training, we use a set of 1,681 face images (collected from Olivetti [171], UMIST [65], Harvard [66], Yale [10] and FERET [141] databases) which have wide variations in pose, facial expression and lighting condition (See http://vision.ai.uiuc.edu/mhyang/mls.html for sample images). In the second mixture method, we start with 8,422 nonface examples from 400 images of landscapes, trees, buildings, etc. Although it is extremely difficult to collect a representative set of nonface examples, the bootstrap method [188] is used to include more nonface examples during training. Each face sample is manually cropped and normalized such that it is aligned vertically and its size is 20×20 pixels. To make the detection method less sensitive to scale and rotation variation, 10 face examples are generated from each original sample. The images are produced by randomly rotating the images by up to 15 degrees with scaling between 80% and 120%. This produces 16,810 face samples.

We test both methods on the three sets of images collected by Rowley [161], Sung [188] and ourselves. Table 5.1 shows the detection rates of our methods and the reported results of several detection methods on the test set in [161]. Experimental results on test set 1, which consists

of 125 images (483 faces) excluding 5 images of hand drawn faces, show that our methods have comparable detection performance with other methods, yet with fewer false detects. Table 5.2

Table 5.1: Experimental results on images from test set 1 (125 images with 483 faces) in [161]

MFA		FLD		Schneiderman [174]		Rowley [161]		Colmenarez [30]	
Detection Rate	False Detects	Detection Rate	False Detects	Detection Rate	False Detects	Detection Rate	False Detects	Detection Rate	False Detects
92.3%	82	93.6%	74	93.0%	88	92.5%	862	93.9%	8122

shows the our experimental results on the test set of Sung and Poggio [188] which consists of 20 images excluding 3 images of line drawn faces (136 faces). Both of our methods consistently perform well and have few false detects. Test set 3 consists of 80 images (252 faces), collected from

Table 5.2: Experimental results on images from test set B (20 images with 136 faces) in [188]

MFA		FLD		Schneiderman [174]		Rowley [161]		Sung [188]		Osuna [136]	
Detection Rate	False Detects	Detection Rate	False Detects	Detection Rate	False Detects	Detection Rate	False Detects	Detection Rate	False Detects	Detection Rate	False Detects
89.4%	3	91.5%	1	89.8%	3	76.8%	5	79.9%	5	74.2%	20

the World Wide Web, with different poses, expressions and faces with heavy shadows (available at http://vision.ai.uiuc.edu/mhyang/mls.html). The detection rates are 86.7% and 88.2% for MFA and FLD-based methods. The number of false detects are 45 and 40, respectively. Both methods perform equally well in detecting these faces though the FLD-based method performs slightly better than the first one. Figures 5.7 and 5.8 show the results of our methods on some test images. See the web page mentioned above for more results. Notice that there is a false detect in the upper left corner of the image in Figure 5.7 since one window resembles a face. Also notice that our methods can detect, up to certain degree, profile faces and faces with heavy shadows. However occluded, rotated faces or faces with sunglasses cannot be detected effectively by both methods due to lack of such examples in the training sets. None of the existing detection methods cannot effectively detect these types of faces except one recent method [162] seems to able to detect rotated faces. Nevertheless, this method cannot detect occluded faces or face with heavy shadows.

Figure 5.7: Sample experimental results using mixture of factor analyzers on images from three test sets. Every detected face is shown with an enclosing window.

Figure 5.8: Sample experimental results using mixture of subspaces with Fisher Linear Discriminant on images from three test sets. Every detected face is shown with an enclosing window.

5.6 Discussion and Conclusion

We have described methods using mixture of linear subspaces methods to detect human faces regardless of their poses, facial expressions and lighting conditions. Both methods find better projection than PCA, thereby facilitating classification of face and nonface patterns. The first method fits a mixture of factor analyzers to estimate the density function of face images, and the second method uses Self-Organizing Map to partition the training set into classes and Fisher Linear Discriminant to find optimal projection for classification. Experimental results on three sets of images demonstrate that both methods perform as well as the best algorithms in detecting upright frontal faces, yet with fewer false detects.

The contributions of this work can be summarized as follows. First, we introduce projection methods that perform better than PCA. Consequently, the classification in the linear subspace is better. Second, we apply mixture models such that the linear subspaces can better capture the variations of face patterns. Although some methods [130] [188] have applied mixture model, they use PCA for projection which is not optimal for classification in subspaces. On the other hand, it is not clear how SVM performs in face detection since the study in [136] has applied SVM on a rather small test set with 136 faces. It will be of great interest to compare our methods with SVM on a large test set since SVM aims to find the optimal hyperplane that minimizes the generalization error under the theoretical upper bounds.

Chapter 6

Learning To Recognize 3D Object With SNoW

A learning account for the problem of object recognition is developed within the PAC (Probably Approximately Correct) model of learnability. The proposed approach makes no assumptions on the distribution of the observed objects, but quantifies success relative to its past experience. Most importantly, the success of learning an object representation is naturally tied to the ability to represent it as a function of some intermediate representations extracted from the image.

We evaluate this approach in a large scale experimental study in which the SNoW learning architecture is used to learn representations for the 100 objects in the Columbia Object Image Database (COIL-100). Experimental results show that the SNoW-based method outperforms the SVM-based system in terms of recognition rate and the computational cost involved in learning. Most importantly, SNoW's performance degrades more gracefully when the training data contains fewer views. The empirical results also provide insight into practical and theoretical considerations on view-based methods for 3D object recognition [158] [225].

6.1 Introduction

The role of learning in computer vision research has become increasingly more significant in recent years. Statistical learning theory has had an influence on many applications ranging from classification and object recognition, grouping and segmentation, illumination modeling, scene reconstruction and others. The rising role of learning methods, made possible by significant improvements in computing power and storage, is largely motivated by the realization that explicit modeling of

complex phenomena in a messy world cannot be done without a significant role of learning, both for model and knowledge acquisition, and to support generalization and avoid brittleness. Nevertheless, many statistical and probabilistic learning models require making explicit assumptions, e.g., on the distribution that governs the occurrences of instances in the world. For many visual inference problems such as recognition, categorization and detection, making these assumptions seems unrealistic.

This work develops a distribution free learning theory account to an archetypical visual recognition problem: object recognition. The problem is viewed as that of learning a representation of an object that, given a new image, is used to recognize the target object in it. The learning account is developed within the PAC (Probably Approximately Correct) model of learnability [202]. This framework allows us to (1) quantify success relative to the distribution of the observed objects, without making assumptions on the distribution. (That is, learnability guarantees that objects sampled from the same distribution as the one that governed the experience of the learner will be recognized correctly.) (2) study the theoretical limits of what can be learned from images in terms of the expressivity of the intermediate representation used by the learning process and (3) develop practical algorithmic solutions to the problem and exhibit their superiority over other methods.

Earlier works have discussed the possibility of identifying the theoretical limits of what can be learned from images [176] and found that learning in terms of the raw representation of the images is computationally intractable. Attempts to explain this focused the dependence of learnability on the representation of the object [43] but failed to provide a satisfactory explanation for it, or a practical solution. The approach developed here builds on suggestions made in [105] and relies heavily on the development of a feature efficient learning approach [115] [22]. That is, a learning process capable of learning in domains in which there is a very large number of potential features but any concept of interest actually depends on a fairly small number of those. At the heart of the learnability approach are two assumptions that we abstract as follows:

Representation: There exists a (possibly infinite) collection $\mathcal{M}$ of "explanations" such that an object can be represented as a simple function of polynomially many elements in $\mathcal{M}$.

Procedural: There exists a process that, given an image that is a positive example of the target object O generates, in polynomial time, "explanations" in $\mathcal{M}$ that are present in the image

and such that, with high probability, at least one of them is in the representation of O.

Under these assumptions we prove that there exists an efficient algorithm that can learn a good representation of the object in the sense that, with high probability, it would make correct predictions on future images that contain (or do not contain) the object. Furthermore, we show that under these conditions, the learned representations are robust under realistic noise conditions. A significant non-assumption of our approach is that it has no prior knowledge on the distribution of images nor it is trying to estimate it. Section 6.2 describes this framework in details.

The framework developed here is very general. The *explanations* alluded to above can represent a variety of computational processes and information sources that operate on the image. They can depend on local properties of the image, the relative positions of primitives in the image, and even external information sources or context variables. Thus, the theoretical support given here applies also to an intermediate learning stage in a hierarchical process. The main assumptions of the framework are discussed in Section 6.3.

For this framework to contribute to a practical solution, and given our assumptions, there needs to be a computational approach that is able to learn in the presence of a large number of potential "explanations". We discuss the related work on view-based object recognition and put our work in this context in Section 6.5. The evaluation we provide for this framework relies on the SNoW learning architecture [22] that is used here in a large scale object recognition experiment. The SNoW system (available publicly at `http://L2R.cs.uiuc.edu/~cogcomp.html`) is described in Section 6.4, and the experimental study in Section 6.6.

6.2 Learning Framework

We study learnability within the standard PAC model [202] and the mistake-bound model [115]. In the learning scenario, we are given a concept class $\mathcal{C}$, a class of $\{0,1\}$-valued function over the instance space X with an associated complexity parameter n, and there is some unknown *target concept* $f_T \in \mathcal{C}$ that we are trying to learn. In the *mistake-bound* model, at each learning stage, an example $x \in X$ is presented; the learning algorithm is asked to predict $f_T(x)$ and is then told whether the prediction was correct. Each time the learning algorithm makes an incorrect prediction, we charge it one *mistake*. We say that $\mathcal{C}$ is *mistake-bound learnable* if there exists a polynomial-time

prediction algorithm $\mathcal{A}$ (possibly randomized) that for all $f_T \in \mathcal{C}$ and any sequence of examples is guaranteed to make at most polynomially many (in n) mistakes. We say that $\mathcal{C}$ is *expected mistake-bound learnable* if there exists $\mathcal{A}$, as above, such that the expected number of mistakes it makes for all $f_T \in \mathcal{C}$ and *any* sequence of examples is at most polynomially many (in n). Note that the expectation is taken over the random choices made by $\mathcal{A}$; there is no probability distribution associated with the sequences. In learning an unknown target function $f_T \in \mathcal{C}$ in the PAC model, we assume that there is a fixed but arbitrary and *unknown* distribution D over the instance space X. The learning algorithm sees examples drawn independently according to D together with their labeling (positive/negative). Then it is required to predict the value of f_T on another example drawn according to D. Denote by $h(x)$ the prediction of the algorithm on the example $x \in X$. The error of the algorithm with respect to f_T and D is measured by $error(h) = Pr_{x \in D}\{f_T(x) \neq h(x)\}$.

We say that $\mathcal{C}$ is *PAC-learnable* if there exists a polynomial-time learning algorithm $\mathcal{A}$ and a polynomial $p(\cdot, \cdot, \cdot)$ such that for all $n \geq 1$, all target concepts $f_T \in \mathcal{C}$, all distribution D over X, and all $\epsilon > 0$ and $0 < \delta \leq 1$, such that if the algorithm $\mathcal{A}$ is given $p(n, 1/\epsilon, 1/\delta)$ examples, then with probability at least $1 - \delta$, $\mathcal{A}$'s hypothesis, h, is such that $error(h) \leq \epsilon$. It can be shown that if a concept class $\mathcal{C}$ is learnable in the expected mistake-bound model (and thus in the mistake bound model) then it is PAC-learnable [70]. In practice, learning is evaluated on a training set. The hope that a classifier learned from a training set will perform well on previously unseen examples is based on the basic theorem of learning theory [202] [203] which, stated informally, guarantees that if the training data and the test data are sampled from the same distribution, good performance on large enough[1] training sample guarantees good performance on the test data (i.e., good "true" error).

6.2.1 Learning Scenario

Let $\mathcal{I}$ be an input space of images. Our goal is to learn a definition `apple`:$\mathcal{I} \rightarrow \{0, 1\}$ that, when evaluated on a given image, outputs 1 when there is an apple in the image, and 0 otherwise. It is clear, though, that this target function is very complex in terms of the input space and, in particular, may depend on relational information and even quantified predicates. Many years of research in

[1] In this sense, the evaluation on a small training set done here is not as optimal as, say, a face detection study [226]. Note, however, that the theory implies that learning scales well with the size of the training data, as we show in Section 6.6.

learning theory, however, have shown that efficient learnability of complex function is not feasible. In the learning scenario described here, therefore, learning will not take effect directly in terms of the raw input. Rather, we will learn the target definitions in terms of an intermediate representation that will be generated from the input image. This will allow us to quantify learnability in terms of the expressivity of the intermediate representation as well as the function learned on top of it and, in particular, it would make explicit the requirements from an intermediate representation so that learning is possible.

Let $\mathcal{I}$ be the *instance space* (e.g., the space of all images). A *relation*[2] is a function $\chi : \mathcal{I} \to \{0, 1\}$. χ can be viewed as an indicator function over $\mathcal{I}$, defining the subset of those elements which are mapped to 1 by χ. A relation χ is *active* in $I \in \mathcal{I}$ if $\chi(I) = 1$.

Definition 1 *Let $\mathcal{X}$ be an enumerable collection of relations on $\mathcal{I}$. A relation-generation function (RGF) is a mapping $G : \mathcal{I} \to 2^{\mathcal{X}}$ that maps $I \in \mathcal{I}$ to a set of all elements in $\mathcal{X}$ that satisfy $\chi(I) = 1$. If there is no $\chi \in \mathcal{X}$ for which $\chi(I) = 1$, $G(I) = \phi$.*

RGFs can be thought as a way to define "kinds" of relations, or to parameterize over a large space of relations. Only when presented with an instance $I \in \mathcal{I}$, a concrete relation(s) is generated. For example, G may be the RGF that corresponds to all vertical edges of size 3 in an image. Given an image, a list of all these edges that are present in the image is produced.

We now present a *mistake-bound* algorithm for a class of functions that can be represented as DNF formulae over the space $\mathbf{X}$ of all relations. As indicated, this implies a PAC learning algorithm, but the proof is for the mistake-bound case is simpler. In Section 6.6 we will learn a more general function – a linear threshold function over conjunctions of relations in $\mathbf{X}$; the theoretical results can be expended to this case.

Definition 2 *Let $\mathcal{X}$ be a set of relations that can be generated by a set of RGFs. Let $\mathcal{M}$ be a collection of monomials (conjunctions) over the elements of $\mathbf{X}$, and $p(n)$, $q(n)$ and $g(n)$ be polynomials. Let $\mathcal{C}_{\mathcal{M}}$ be the class of all functions which are disjunctions of at most $p(n)$ monomials in $\mathcal{M}$. Following [105], we call $\mathcal{C}_{\mathcal{M}}$* polynomially explainable *if there exists an efficient (polynomial-time) algorithm $\mathcal{B}$ such that for every function $f \in \mathcal{C}_{\mathcal{M}}$, and every positive example of f as input,*

[2] In the machine learning literature a relation is sometimes called a feature.

$\mathcal{B}$ outputs at most $q(n)$ monomials (not necessarily all of them are in $\mathcal{M}$) such that with probability at least $1/g(n)$ at least one of them appears in f (the probability is taken over the coin-flips of the (possibly probabilistic) algorithm $\mathcal{B}$).

We note that, in principle, it is possible to abstract the generation of the conjunctions into the RGFs (Def. 1). However, we would like to emphasize the generation of conjunction over simple relations and the possibility of learning on top of it, given arguments in the literature of its effectiveness and potential biological plausibility [51] [200].

We emphasize that f itself is *not* given to the algorithm $\mathcal{B}$. Also note that a function f in the class $\mathcal{C}_{\mathcal{M}}$ may have several equivalent representations over $\mathcal{M}$.

Theorem 1 *If $\mathcal{C}_{\mathcal{M}}$ is polynomially explainable then $\mathcal{C}_{\mathcal{M}}$ is expected mistake-bound learnable. Furthermore, if $\mathcal{C}_{\mathcal{M}}$ is polynomially explainable by an algorithm $\mathcal{B}$ that always outputs at least one term of f (i.e., $g(n) \equiv 1$) then $\mathcal{C}_{\mathcal{M}}$ is mistake-bound learnable.*

Proof: [sketch] The algorithm is similar to an algorithm presented in [13] which learns a disjunction of terms in the infinite attribute model. The algorithm maintains an hypothesis h which is a disjunction of monomials. Initially h contains no monomials (i.e., $h \equiv FALSE$). Upon receiving an example e, the algorithm predicts $h(e)$; if the prediction is correct, h is not updated. Otherwise, upon a mistaken prediction, it proceeds as follows:

- If e is positive: execute $\mathcal{B}$ (the algorithm guaranteed by the assumption that $\mathcal{C}_{\mathcal{M}}$ is polynomially explainable) on the example e and add the monomials it outputs to h.
- If e is negative: remove from the hypothesis h all the monomials that are satisfied by e (there must be at least one).

The analysis of the algorithm is straight forward for the case $g(n) \equiv 1$, and more subtle in general. ■

The algorithm used in practice, in SNoW, is conceptually similar. The main difference is that the hypothesis h is a linear threshold function over elements in $\mathcal{M}$, and rather than dropping elements from it, their weight is updated. The details of this process (Section 6.4) are crucial for our approach to be computationally feasible for large scale domains and for robustness.

6.2.2 Robustness

Any realistic learning framework needs to support different kinds of noise in its input. Several kinds of noise have been studied in the literature in the context of PAC learning, and algorithms of the type we consider here have been shown to be robust to them. The most studied type of noise is that of classification noise [92] in which the examples are assumed to be given to the learning algorithm with labels that are flipped with some probability, smaller than 1/2. Learning in our framework can be shown to be robust to this kind of noise, as well as to a more realistic case of *attribute noise*, in which the the description of the input itself is corrupted to a certain degree. We believe that this is the type of noise that is more relevant in the current case. First, since learning is done in terms of the output of the RGFs, which may introduce some noise. Second, since attribute noise is related to *occlusion* noise that is important in object recognition. Specifically, attribute noise can be used to model the type of noise that usually occurs when other objects appear in the image, behind or in front of the target object. This is formalized using the notion of *domination*: Let f_1, f_2 be two concepts. We say that f_1 is *k-dominated* by f_2 if each f_1 example can be obtained from an f_2 example by flipping at most k of the active relations. In this case, f_2 *k-dominates* f_1. The labels of the examples, however, are generated according to the original concept, before the noise is introduced.

Theorem 2 *If a class $\mathcal{C}_{\mathcal{M}}$ is learnable by virtue of being polynomially explainable then it is learnable even if examples of the target class are cluttered by a k-domination attribute noise, for any constant k.*

Proof: [Sketch] The proof is an extension of the arguments in [116] regarding robustness to attribute noise, to the case of the infinite attribute model. ■

6.3 From Theory to Practice

Several issues need to be addressed in order to exhibit the practicality of our learning framework. The first is the availability of a variety of RGFs, capable of extracting from data evidence for the existence of primitive visual patterns under different conditions. In this work we illustrate the approach by utilizing simple edge detectors. The second issue is the composition of complex

relations from primitive ones. This is crucial since it allows the representation of complex functions in terms of the relations by learning simple functional descriptions over their compositions. A language that supports composition of restricted families of conjunctions and can encode structural relations in images (e.g., above, to the left of...) is discussed in a companion paper. This work uses only general conjunctions and restricts only their size.

Finally, the issue of learnability which is the focus of this work. In learning situations in vision, the number of potential relations (features) that may affect each decision is very large but, typically, only a small number of them is actually relevant to a decision. Beyond correctness, a realistic learning approach needs therefore to be relation-efficient [115] in that its learning complexity (number of examples required for convergence) depends on the number of relevant relations and not the global number of relations in the domain. This can be phrases as the dependence of generalization quality on the number of examples observed. Also, since only a few of the many potential relations are active in any instance, the complexity of evaluating the learning hypothesis on an instance should depend on the number of active relations in the input rather than the total number in the domain. And, a learning approach should allow variable input size, since learning is in terms of relations that are generated from the image in a data-driven way, making it impossible, or impractical, for a learning approach to write explicitly, in advance, all possible "relations".

Given that, the learning approach used in this work is the one developed within the SNoW learning architecture [157] [22]. SNoW is specifically tailored for learning in domains in which the potential number of features taking part in decisions is very large, but may be unknown a priori, as in the infinite attribute learning model [13]. Specifically, as input, the algorithm receives labeled instances $< (x, l) >$, where an instance $x \in \{0, 1\}^{\infty}$ is presented as a list of all the *active* relations in it and the label is a member of a discrete set of values (e.g., objects identifiers). Given a domain instance (an image) a set of pre-existing RGFs are evaluated on it and generate a collection of relations that are active in this image; these, in turn, may be composed to generate the elements of $\mathcal{M}$. A list of active elements in $\mathcal{M}$ is presented to the learning procedure and learning is done at this level.

6.4 The SNoW Architecture

The SNoW (Sparse Network of Winnows[3]) learning architecture is a sparse network of linear units over a common pre-defined or incrementally learned feature space. Nodes in the input layer of the network represent simple relations over the input instance and are being used as the input features. Each linear unit is called a *target node* and represents relations or concepts which are of interest over the input; in the current application, target nodes represent a definition of an object in terms of the relations (features) extracted from the 2D image input. An input instance is mapped into a set of features which are active in it; this representation is presented to the input layer of SNoW and propagates to the target nodes. Target nodes are linked via weighted edges to (some of) the input features.

Let $\mathcal{A}_t = \{i_1, \ldots, i_m\}$ be the set of features that are active in an example and are linked to the target node t. Then the linear unit corresponding to t is *active* iff

$$\sum_{i \in \mathcal{A}_t} w_i^t > \theta_t,$$

where w_i^t is the weight on the edge connecting the ith feature to the target node t, and θ_t is the threshold for the target node t.

Each SNoW *unit* may include a collection of subnetworks, one for each of the target relations but all using the same feature space. In the current case, we may have one unit with target subnetworks for all the target objects or we may define different units, each with two competing target objects. A given example is treated autonomously by each target subnetwork; an example labeled t may be treated as a positive example by the subnetwork for t and as a negative example by the rest of the target nodes.

The learning policy is on-line and mistake-driven; several update rules can be used within SNoW. The most successful and the only one used in this work, is a variant of Littlestone's Winnow update rule [115], a multiplicative update rule that we tailored to the situation in which the set of input features is not known a priori, as in the infinite attribute model [13]. This mechanism is implemented via the sparse architecture of SNoW. That is, (1) input features are allocated in a

[3]To winnow: to separate chaff from grain.

data driven way – an input node for the feature i is allocated only if the feature i was active in any input sentence and (2) a link (i.e., a non-zero weight) exists between a target node t and a feature i if and only if i was active in an example labeled t.

One of the important properties of the sparse architecture is that the complexity of processing an example depends only on the number of features active in it, n_a, and is independent of the total number of features, n_t, observed over the life time of the system. This is important in domains in which the total number of features is very large, but only a small number of them is active in each example.

The Winnow update rule has, in addition to the threshold θ_t at the target t, two update parameters: a *promotion* parameter $\alpha > 1$ and a *demotion* parameter $0 < \beta < 1$. These are being used to update the current representation of the target t (the set of weights w_i^t) only when a mistake in prediction is made. Let $\mathcal{A}_t = \{i_1, \ldots, i_m\}$ be the set of active features that are linked to the target node t. If the algorithm predicts 0 (that is, $\sum_{i \in \mathcal{A}_t} w_i^t \leq \theta_t$) and the received label is 1, the active weights in the current example are *promoted* in a multiplicative fashion:

$$\forall i \in \mathcal{A}_t, w_i^t \leftarrow \alpha \cdot w_i^t.$$

If the algorithm predicts 1 ($\sum_{i \in \mathcal{A}_t} w_i^t > \theta_t$) and the received label is 0, the active weights in the current example are *demoted*:

$$\forall i \in \mathcal{A}_t, w_i^t \leftarrow \beta \cdot w_i^t.$$

All other weights are unchanged.

The key feature of the Winnow update rule is that the number of examples required to learn a linear function grows linearly with the number n_r of *relevant* features and only logarithmically with the total number of features. This property seems crucial in domains in which the number of potential features is vast, but a relatively small number of them is relevant. Moreover, in the sparse model, the number of examples required before converging to a linear separator that separates the data (provided it exists) scales with $O(n_r \log n_a)$. Winnow is known to learn efficiently any linear threshold function and to be robust in the presence of various kinds of noise and in cases where no linear-threshold function can make perfect classifications, while still maintaining its abovementioned

dependence on the number of total and relevant attributes [116] [99].

Once target subnetworks have been learned and the network is being evaluated, a decision support mechanism is employed, which selects the dominant active target node in the SNoW unit via a winner-take-all mechanism to produce a final prediction. In other applications the decision support mechanism may also cache the output and process them along with the output of other SNoW units to produce a coherent output.

Figures 6.1, 6.2 and 6.3 provide more details on the SNoW learning architecture.

SNoW: *Objects and Notation*

$F = \mathcal{Z}^+ = \{0, 1, \dots\}$	/* Set of potential features */
$T = \{t_1, \dots t_k\} \subset F$	/* Set of targets */
$F_t \subseteq F$	/* Set of features linked to target t */
$t_{NET} = \{[(i, w_i^t) : i \in F_t], \theta_t\}$	/* The representation of the target t. */
$activation : T \rightarrow \Re$	/* activation level of a target t. */
$SNoW = \{t_{NET} : t \in T\}$	/* The SNoW Network */
$e = \{i_1, \dots, i_m\} \subset F^m$	/* An example, represented as a list of active features */

Figure 6.1: SNoW: *Objects and Notation*

SNoW: *Training and Evaluation*

Training Phase: SNoW-Train (SNoW, e)

Initially: $F_t = \phi$, for all $t \in T$.

For each $t \in T$

1. UpdateArchitecture (t, e)
2. Evaluate (t, e)
3. UpdateWeights (t, e)

Evaluation Phase: SNoW-Evaluation(SNoW, e)

For each $t \in T$

Evaluate (t, e)

MakeDecision (SNoW, e)

Figure 6.2: SNoW: *Training and Evaluation.* Training is the learning phase in which the network is constructed and weights are adjusted. Evaluation is the phase in which the network is evaluated, given an observation. This is a conceptual distinction; in principle, one can run in on line mode, in which training is done continuously, even when the network is used for evaluating examples.

SNoW: *Building Blocks*

Procedure Evaluate(t, e)

activation = $\sum_{i \in e} w_i^t$

Procedure UpdateWeights(t, e)

If $(activation(t) > \theta_t)$ & $(t \notin e)$ /* predicted positive on negative example */

for each $i \in e$: $w_t^i \leftarrow w_t^t \cdot \beta$

If $(activation(t) \leq \theta_t)$ & $(t \in e)$ /* predicted negative on a positive example */

for each $i \in e$: $w_t^i \leftarrow w_t^t \cdot \alpha$

Procedure UpdateArchitecture(t, e)

If $t \in e$

- For each $i \in e \setminus F_t$, set $w_i^t = w$ /* Link feature to target; set initial weight */

Otherwise: do nothing

Procedure MakeDecision(SNoW, e)

Predict winner = $argmax_{t \in T} activation(t)$ /* Winner-take-all Prediction */

Figure 6.3: SNoW: *Main Procedures*

Applying SNoW to 3D object recognition requires specifying the architecture used and the representation chosen for the input images. As described above, to perform object recognition we associate a target subnetwork with each target object. This target learns a definition of the object in terms of the input features extracted from the image. We could either define a single SNoW unit which contains target subnetworks for all the 100 different target objects, or we may define different units, each with several (e.g., two) competing target objects. Selecting a specific architecture makes a difference both in training time, where learning a definition for object a makes use of negative examples of other objects that are part of the same unit but, more importantly, it makes a difference in testing; rather that two competing objects for a decision, there may be a hundred. The chances for a spurious mistake caused by an incidental view point are clearly much higher. On the other hand, it has significant advantages in terms of space complexity and the appeal of the evaluation mode. This point will be discussed later.

An SVM is a two-class classifier which, for an n-class pattern recognition problem, trains $\frac{n(n-1)}{2}$ binary classifiers. Since we compare the performance of the proposed SNoW-based method with SVMs, in order to maintain a fair comparison we have to perform it in the *one-against-one* scheme.

That is, we use SNoW units of size two. To classify a test instance, tournament-like pair-wise competition between all the machines is performed and the winner determines the label of the test instance. The recognition rates of the SVM and SNoW based methods shown in Table 6.2 were performed using the one-against-one scheme. (That is, we trained $\binom{100}{2} = 4,950$ classifiers for each method and evaluated $98 (= 50 + 25 + 12 + 6 + 3 + 1 + 1)$ classifiers on each test instance.

6.5 View-Based Methods

The appearance of an object is the combined effects of its shape, reflectance properties, pose, and the illumination in the scene. While shape and reflectance are intrinsic properties that do not change for a rigid object, pose and illumination vary from one scene to another. View-based recognition methods attempt to use data observed under different poses and illumination conditions to learn a compact model of the object's appearance; this, in turn, is used to resolve the recognition problem from view points that were not observed previously.

A number of view-based schemes have been developed to recognize 3D objects. Poggio and Edelman [145] show that 3D objects can be recognized from the raw intensity values in 2D images (we call this representation here a *pixel-based representation*) using a network of generalized radial basis functions. They argue and demonstrate that full 3D structure of an object can be estimated if enough 2D views of the object are provided. Turk and Pentland [199] demonstrate that human faces can be represented and recognized by "eigenfaces." Representing a face image as a vector of pixel values, the eigenfaces are the eigenvectors associated with the largest eigenvalues which are computed from a covariance matrix of the sample vectors. An attractive feature of this method is that the eigenfaces can be learned from the sample images in pixel representation without any feature selection. The eigenspace approach has since been used in different vision tasks from face recognition to object tracking. Murase and Nayar [131] [132] develop a parametric eigenspace method to recognize 3D objects directly from their appearance. For each object of interest, a set of images in which the object appears in different poses is obtained as training examples. Next, the eigenvectors are computed from the covariance matrix of the training set. The set of images is projected to a low dimensional subspace spanned by a subset of eigenvectors, in which the object is represented as a manifold. A compact parametric model is constructed by interpolating the

points in the subspace. In recognition, the image of a test object is projected to the subspace and the object is recognized based on the manifold it lies on. Using a subset of the Columbia Object Image Library (COIL-100), they show that 3D objects can be recognized accurately from their appearances in real-time.

In contrast to these algebraic methods, general purpose learning methods such as support vector machines (SVMs) have also been used for this problem. Schölkopf [175] was the first to apply SVMs to recognize 3D objects from 2D images and has demonstrated the potential of this approach in visual learning. Pontil and Verri [147] also used SVMs for 3D object recognition and experimented with a subset of the COIL-100 dataset. Their training set consisted of 36 images (one for every 10°) for each of the 32 objects they chose, and the test sets consist of the remaining 36 images for each object. For 20 random selections of 32 objects from the COIL-100, the system achieves perfect recognition rate (but see comments on that in Section: 6.6). More recently, a subset of the COIL-100 has been used by also Roobaert and Van Hulle [156] to compare the performance of SVMs with different pixel-based input representations.

Given the success of this approach, which we use to compare with the approach presented here, we present below the SVM method in some more details.

6.5.1 Support Vector Machines

Support Vector Machine (SVM) [203] [32] is a general purpose learning method for pattern recognition and regression problems that is based on the theory of structural risk minimization. According to the structural risk minimization inductive principle, a function that describes the training data well and belongs to a set of functions with low VC dimension[4] will generalize well (that is, will guarantee a small expected recognition error for the unseen data points) regardless of the dimensionality of the input space [203]. Based on this principle, the SVM is a systematic approach to find a linear function (a hyperplane) that belongs to a set of functions of this forms with the lowest VC dimension. The reason for using a linear function is that for a set of linearly separable points, it is possible to explicitly quantify the VC dimension in terms of the minimal distance between positive and negative points. SVMs provide non-linear function approximations by mapping the input vec-

[4]The VC dimension of a class of functions is a combinatorial parameter that measures the richness of the function class. See [203] [93] for details.

tors into a high dimensional feature space where a linear hyperplane that separates the data exists. It can also be extended to cases where the best hyperplane in the resulting high dimension space does not quite separate all the data points.

Given a set of samples $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_l, y_l)$ where $\mathbf{x}_i \in R^N$ is the input vector and $y_i \in \{-1, 1\}$ its label, an SVM aims to find an optimal hyperplane that leaves the largest possible fraction of data points of the same class on the same side while maximizes the distance of either class from the hyperplane (margin distance). Vapnik [203] shows that maximizing the margin distance is equivalent to minimizing the VC dimension and therefore contributes to better generalization. The problem of finding the optimal hyperplane is thus posed as a constrained optimization problem and solved using quadratic programming techniques. The optimal hyperplane, which determines the class label of a data point $\mathbf{x} \in R^N$, is of the form

$$f(\mathbf{x}) = sgn(\sum_{i=1}^{l} y_i \alpha_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b)$$

where $k(\cdot, \cdot)$ is a kernel function and sgn is the function that outputs $+1$ on positive inputs and -1 otherwise. Constructing an optimal hyperplane is equivalent to determining the nonzero α_is. Sample vectors $\mathbf{x}_i$ that corresponds to a nonzero α_i are called the *support vector*s (SVs) of the optimal hyperplane. The hope, when using this method, is for a small number of support vectors, thereby producing a compact classifier.

The use of kernel functions allows, using Mercer theorem, to avoid the need to blow up the dimensionality in order to reach a state in which the sample is linearly separable. If the kernel is of the form

$$k(\mathbf{x}, \mathbf{x}_i) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i)$$

for some nonlinear function $\Phi : R^N \rightarrow F^M, \quad M \gg N$ the computation can be done in the original, lower dimension space rather than working in the M dimensional space, although the hypoerplane is constructed in R^M. For a linear SVM, the kernel function is simply the dot product of vectors in the input space. Several kernel functions, such as polynomial functions and radial basis functions, have been shown to satisfy Mercer theorem and used in nonlinear SVM, allowing the construction

of a variety of learning machines, some of which coincide with classical architectures. However, this also results in a drawback since one needs to find the "right" kernel function when using SVMs. It is interesting to observe, though, that although the use of kernel functions seems to be one of the advantages of SVMs from a theoretical point of view, most experimental studies have used linear SVMs which were found to perform better. One potential reason is that SVMs are prone to outliers and various kinds of noise in the data, and this gets worse when non-linear kernels are used.

6.6 Experimental Evaluation

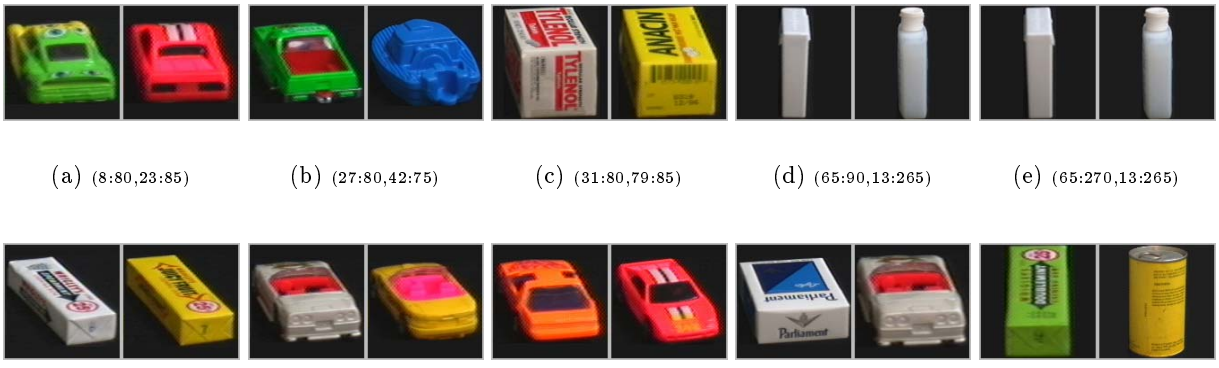
We use the Columbia Object Image Library (COIL-100) database in all the experiments below. COIL is available at http://www.cs.columbia.edu/CAVE. The COIL-100 dataset consists of color images of 100 objects where the images of the objects that were taken at pose intervals of 5°, i.e., 72 poses per object. The images were also normalized such that the larger of the two object dimensions (height and width) fits the image size of 128×128 pixels. Figure 6.4 shows the images of the 100 objects taken in frontal view, i.e., zero pose angle. The 32 highlighted objects in Figure 6.4 are considered more difficult to recognize in [147]; we use all 100 objects including these in our experiments. Each color image is converted to a gray-scale image of 32×32 pixels for our experiments.

Figure 6.4: Columbia Object Image Library (COIL-100) consists of 100 objects of varying poses 5° apart). The objects are shown in row order where the highlighted ones are those considered more difficult to recognize in [147].

6.6.1 Ground Truth of the COIL-100 Dataset

At first glance, it seems difficult to recognize the objects in the COIL dataset because it consists of a large number of objects with varying pose, texture, shape and size. Since each object has 72 images of different poses (5° apart), many view-based recognition methods use 36 (10° apart) of them for training and the remaining images for testing. However, it turns out that under these dense sampling conditions the recognition problem is not difficult (even when only grey-level images are used). Namely, in this case, instances that belong to the same object are very close to each other in the image space (where each data point represents an image of an object in a certain pose). We verified this by experimenting with a simple nearest neighbor classifier (using the Euclidean distance), resulting in an average recognition rate of 98.50% (54 errors out of 3,600 tests). Figure 6.6.1 shows some of the objects misclassified by nearest neighbor method.

In principle, one may want to avoid using the nearest neighbor method since it requires a lot of memory for storing templates and its recognition time complexity is high. The goal here was simply to show that this simple method is comparable to the complex SVM approaches [147] [156] for the case of dense sampling. Therefore, the abovementioned recognition problem is not appropriate for comparison among different methods.



(a) (8:80,23:85) (b) (27:80,42:75) (c) (31:80,79:85) (d) (65:90,13:265) (e) (65:270,13:265)

(f) (67:70,84:75) (g) (69:80,91:75) (h) (76:80,23:255) (i) (96:260,69:85) (j) (98:270,99:195)

Figure 6.5: Mismatched objects using the nearest neighbor method. $(x : a, y : b)$ means that object x with view angle a is recognized as object y with view angle b. It shows some of the 54 errors (out of 3,600 test samples) made by the nearest neighbor classifier when there are 36 views per object in the training set.

Table 6.1: Recognition rates of nearest neighbor classifier

	30 objects randomly selected from COIL	32 objects shown in Figure 6.4 selected by [147]	The whole 100 objects in COIL
Errors/Tests	14/1080	46/1152	54/3600
Recognition rate	98.70%	96.00%	98.50%

It is interesting to see that the pairs of the objects on which the nearest neighbor method misclassified have similar geometric configurations and similar poses. A close inspection shows that most of the recognition errors are made between the three packs of chewing gums, bottles and cars. Other dense sampling cases are easier for this method. Consequently, the set of selected objects in an experiment has direct effects on the recognition rate. This needs to be taken into account when evaluating results that use only a subset of the 100 objects (typically 20 to 30) from the COIL dataset for experiments. Table 6.1 shows the recognition rates of nearest neighbor classifiers in several experiments in which 36 poses of each object are used for templates and the remaining 36 poses are used for tests.

Given this baseline experiment we have decided to perform our experimental comparisons in cases in which the number of views of objects available in training is limited. Some of our preliminary results were presented in [224] [225].

6.6.2 Experiment Setups

Applying SNoW to 3D object recognition requires specifying the architecture used and the representation chosen for the input images. As described above, to perform object recognition we associate a target subnetwork with each target object. This target learns a definition of the object in terms of the input features extracted from the image. We could either define a single SNoW unit which contains target subnetworks for all the 100 different target objects, or we may define different units, each with several (e.g., two) competing target objects. Selecting a specific architecture makes a difference both in training time, where learning a definition for object a makes use of negative examples of other objects that are part of the same unit but, more importantly, it makes a difference in testing; rather that two competing objects for a decision, there may be a hundred.

The chances for a spurious mistake caused by an incidental view point are clearly much higher. On the other hand, it has significant advantages in terms of space complexity and the appeal of the evaluation mode. This point will be discussed later.

An SVM is a two-class classifier which, for an n-class pattern recognition problem, trains $\frac{n(n-1)}{2}$ binary classifiers. Since we compare the performance of the proposed SNoW-based method with SVMs, in order to maintain a fair comparison we have to perform it in the *one-against-one* scheme. That is, we use SNoW units of size two. To classify a test instance, tournament-like pair-wise competition between all the machines is performed and the winner determines the label of the test instance. The recognition rates of the SVM and SNoW based methods shown in Table 6.2 were performed using the one-against-one scheme. (That is, we trained $\binom{100}{2} = 4,950$ classifiers for each method and evaluated $98 (= 50 + 25 + 12 + 6 + 3 + 1 + 1)$ classifiers on each test instance.

6.6.3 Results Using Pixel-Based Representation

Table 6.2 shows the recognition rates of the SNoW-based method, the SVM-based method (using linear dot product for the kernel function), and the nearest neighbor classifier using the COIL-100 dataset. The important parameter here is that we vary the number of views of an object (n) during training and use the rest of the views ($72 - n$) of an object for testing.

Table 6.2: Experimental results of three classifiers using the 100 objects in the COIL-100 dataset

	# of views/object			
	36	18	8	4
	3600 tests	5400 tests	6400 tests	6800 tests
SNoW	95.81%	92.31%	85.13%	81.46%
Linear SVM	96.03%	91.30%	84.80%	78.50%
Nearest Neighbor	98.50%	87.54%	79.52%	74.63%

The experimental results show that the SNoW-based method performs as well as the SVM-based method when many views of the objects are present during training and outperforms SVM-based method when the numbers of views is limited. Although it is not surprising to see that the recognition rate decreases as the number of views available during training decreases, it is worth noticing that both SNoW and SVM are capable of recognizing 3D objects in the COIL-100 dataset

with satisfactory performance if enough views (e.g., > 18) are provided. Also they seems to be fairly robust even if only a limited number of views (e.g., 8 and 4) are used for training; the performance of both methods degrades gracefully.

To provide some more insight into these methods, we note that in the SVM-based methods, only 27.78% (20 out of 72) of the input vectors serves as support vectors. For SNoW, out of 262,144 potential features in the pixel-based representation, only 13,805 were active in the dense case (i.e., 36 views). This shows the advantage gained from using the sparse architecture. However, only a small number of those may be relevant to the representation of each target, as a more careful look as the SNoW output hypothesis reveals.

An additional potential advantage of the SNoW architecture is that it does not learn discriminators, but rather can learn a representation for each object, which can then be used for prediction in the one-against-all scheme or to build hierarchical representations. However, as is shown in Table 6.3, this implies a significant degradation is the performance. Finding a way to make better predictions in the one-against-all scheme is one of the important issues for future investigation, to better exploit the advantages of this approach.

Table 6.3: Recognition rates of SNoW using two learning paradigms

SNoW	# of views/object			
	36	18	8	4
one-against-one	95.81%	92.31%	85.13%	81.46%
one-against-all	90.52%	84.50%	81.85%	76.00%

6.6.4 Results Using Edge-Based Representation

For each 32×32 edge map, we extract horizontal and vertical edges (of length at least 3 pixels) and then encode as our features conjunctions of two of these edges. The number of potential features of this sort is $\binom{2048}{2} = 2,096,128$. However, only an average of 1,822 of these is active for objects in the COIL-100 dataset. To reduce the computational cost the feature vectors were further pruned and only the 512 most frequently occurring features were retained in each image.

Table 6.4 shows the performance of the SNoW-based method when conjunctions of edges are used to represent objects. As before, we vary the number of views of an object (n) during training

and use the rest of the views $(72-n)$ of an object for testing. The results indicate that conjunctions of edges provide useful information for object recognition and that SNoW is able to learn very good object representations using these features. The experimental results also exhibit the relative advantage of this representation increases when the number of views per object is limited.

Table 6.4: Experimental results of three classifiers using the 100 objects in the COIL-100 dataset

	# of views/object			
	36	18	8	4
	3600 tests	5400 tests	6400 tests	6800 tests
SNoW w/ conjunction of edges	**96.25%**	**94.13%**	**89.23%**	**88.28%**
SNoW w/ intensity values	95.81%	92.31%	85.13%	81.46%
Linear Support Vector Machine	96.03%	91.30%	84.80%	78.50%
Nearest Neighbor	98.50%	87.54%	79.52%	74.63%

6.6.5 Noise Model

To test whether the proposed learning framework is noise tolerant, we select a set of 10 objects[5] from the COIL-100 dataset and add in artificial occlusions for experiments. In the data set, each object has 36 images (10° apart) for training and the remaining 36 images for tests. The object images are occluded by a strip controlled by four parameters (α, p, l, g) where α denote the angle of the strip, p denote the percentage of occluded area, l denote the location of the center of the strip, and g denote the intensity values of the strip. Figure 3 shows some object images and the occluded object images for $\{\alpha, p, l, g\} = \{45^\circ, 15\%, (16, 16), 0\}$. This SNoW classifier is tested against this

(a) (b)

Figure 6.6: Objects images with and without occlusion as well their edge maps.

[5]More specifically, the objects are selected from the set of objects on which the nearest neighbor classifier makes most mistakes, objects 8, 13, 23, 27, 31, 42, 65, 78, 80, 91.

dataset using the edge-based representation. Table 6.2 shows the experimental results with and without occlusions on this set of 10 objects with 36 views. The recognition performance degrades only slightly from 92.03% to 88.78%. Note that the objects are those on which the nearest neighbor classifier makes most mistakes.

Table 6.5: Experimental results of SNoW classifier on occluded images with 36 views per object

	Recognition rate w/o occlusion	Recognition rate w/ occlusion
SNoW	92.03%	88.78%

6.7 Discussion and Conclusion

We proposed a learning framework for visual learning and evaluated it experimentally in the context of learning for object recognition. In this approach learnability can be rigorously studied without making assumptions on the distribution of the observed objects but, via the PAC model, the learned hypothesis' performance naturally depends on its prior experience.

An important feature of the approach is that learning is not studied directly in terms of the raw data but rather with respect to intermediate representations extracted from it and can thus be quantified in terms of the ability to generate expressive intermediate representations. It particular, it makes explicit the requirements from these representations to allow learnability. We believe that research in vision should concentrate on the study of these intermediate representations.

For a fair comparison among different methods, this chapter uses pixel-based presentation in the experiments. However, we view the edge-based representation that was found to be even more effective and robust as another starting point for future research. We believe that pursing the direction of using complex intermediate representations will benefit future work on recognition and, in particular, robust recognition under various types of noise.

We have illustrated our approach in a large scale experimental study in which we use the SNoW learning architecture to learn representations for the 100 objects in COIL-100. Although it is clear that object recognition in isolation is not the ultimate goal, this study shows the potential of this computational approach as a basis for studying and supporting more realistic visual inferences.

Chapter 7

Geometric Approach To Train Support Vector Machines

Support Vector Machines (SVMs) have shown great potential in numerous visual learning and pattern recognition problems. The optimal decision surface of an SVM is constructed from its support vectors which are conventionally determined by solving a quadratic programming (QP) problem. However, solving a large optimization problem is challenging since it is computationally intensive and the memory requirement grows with square of the number the training vectors. In this chapter, we propose a geometric method to extract a small superset of support vectors, which we call guard vectors, to construct the optimal decision surface. Specifically, the guard vectors are found by solving a set of linear programming problems. Experimental results on synthetic and real data sets show that the proposed method is more efficient than conventional methods using QPs and requires much less memory.

7.1 Introduction

Support Vector Machine (SVM) is a novel machine learning algorithm based on statistical learning theory that can be applied to pattern recognition and regression problems [203] [32]. One distinct characteristic of SVMs is that it aims to find the optimal hyperplane from a set training samples such that the expected recognition error for the unseen test samples is minimized. According to the structural risk minimization inductive principle, a function that describes the training data well and belongs to a set of functions with lowest VC (Vapnik-Chervonenkis) dimension will generalize well regardless of the dimensionality of the input space [32]. Based on this principle, SVM adopts

a systematic approach to find a linear function that belongs to a set of functions with lowest VC dimension. The SVM algorithm also provides non-linear function approximations by projecting the input vectors to a high dimensional feature space in which a linear hyperplane is constructed to separate all the projected vectors. One novelty of the nonlinear SVMs is the use of kernel functions to avoid the expensive computations imposed by the nonlinear projection. Although there is no guarantee that the patterns in a high dimensional space can always be linearly separated by a hyperplane, in practice it is usually feasible to find one such linear hyperplane in the projected space.

SVMs have recently attracted much attention because of the rigorous theoretical derivations from statistical learning theory and excellent empirical results in many classification tasks. Training an SVM is equivalent to solving a quadratic optimization problem in which the support vectors (SVs) are identified to construct the optimal hyperplane. However, many researchers have found that SVMs are not only computationally expensive but also memory intensive.

Consider a training set of m examples, $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, $\ldots$, $(\mathbf{x}_m, y_m)$, where $\mathbf{x}_i$ is an input vector and y_i is its class label, one can solve the following quadratic programming (QP) problem to find the optimal hyperplane:

$$\begin{array}{ll} \text{Minimize} & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i Q_{ij}\alpha_j - \sum_{i=1}^{m}\alpha_i \\ \text{Subject to} & 0 \le \alpha_i \le C \ \text{ and } \ \sum_{i=1}^{m} y_i\alpha_i = 0 \end{array}$$

where C is a parameter for soft margin classifier and Q is an $m \times m$ matrix that depends on the training inputs $\mathbf{x}_i$, the label y_i, and the kernel function of an SVM. Note that a training set of 50,000 examples will yield a Q matrix with 2.5 billion elements, which cannot easily fit into the memory of a standard computer. Solving a large QP problem is usually computationally and memory intensive, nontrivial to implement, and can suffer from numerical instability.

Although several methods have been developed to efficiently solve the quadratic optimization problem [136] [144] [90], none use the *structure* information of the training vectors. Since the support vectors (SVs) form a subset of training vectors which have equal minimum perpendicular distance to the optimal hyperplane and the optimal hyperplane is a weighted linear combination of these SVs, we can construct *exactly* the same optimal hyperplane if we are given only the SVs.

In this chapter, we propose a method to extract a superset of the SVs based on the structural information of the training vectors. Note that for each support vector $\mathbf{x}_s$ with label y_s, we can always find a hyperplane that passes through $\mathbf{x}_s$ and separates the vectors with label y_s and those with the opposite class label. In other words, a training vector may be a support vector if there exists such a hyperplane that linearly separates all the vectors according to their labels.

The duality theory in computational geometry specifies that a point or a vector in any dimensional space has a unique corresponding hyperplane in the dual space [148], and vice versa. We are interested in determining whether there exits a hyperplane through a vector $\mathbf{x}_i$ in the primal space such that all the points are linearly separated according to their labels in the primal space. Since all the points in the primal space have dual hyperplanes in the dual space and their normal directions are determined by their labels, the existence of a separating hyperplane at $\mathbf{x}_i$ in the primal space is equivalent to feasibility of a linear program in the dual space. If there exists such a hyperplane, there exists a corresponding point in the dual space. Since the hyperplane through $\mathbf{x}_i$ can linearly separate all the points, $\mathbf{x}_i$ may be a support vector. We will call such vectors the *guard vectors* in the rest of this chapter. For an optimal hyperplane with unit normal $\mathbf{w}$, the support vectors must lie on hyperplanes with normalized distance $\frac{1}{||\mathbf{w}||}$ to the optimal hyperplane. Therefore, every support vector must be a guard vector. However, a guard vector may not be a support vector since it may not have minimum distance to the optimal hyperplane.

For a set of m training examples, the superset of SVs are found by solving a set of m linear programming problems. Each LP aims at determining whether a point can be a support vector on the optimal hyperplane. Note that we are interested in the feasibility of each LP rather than the optimal objective value of that LP. After we extract the set of guard vectors, we can construct the same optimal hyperplane by solving a quadratic programming problem with these vectors. Since the guard vectors usually form a small superset of support vectors and linear programming problems can be solved more efficiently than quadratic programming problems, the proposed method is an efficient algorithm to train SVMs. Our experimental results on several benchmarks show that the proposed method has low time and space complexity.

This chapter is organized as follows. Section 7.2 describes previous approaches to train SVMs. In Section 7.3, we present a method to extract guard vectors using linear programming and then

construct an optimal hyperplane. In Section 7.4, we present experimental results on several synthetic datasets and real data sets. We conclude this chapter with comments on current and future work in Section 7.5.

7.2 Related Work

For large learning problems with many training examples, the constrained quadratic optimization approach to solving SVMs quickly becomes intractable in terms of time and memory requirements. A training set of 50,000 examples will yield a Q matrix with 2.5 billion elements, which cannot easily fit into the memory of a standard computer. Consequently, traditional optimization algorithm such as Newton, Quasi Newton, etc., cannot be directly applied (since these methods usually involve the whole Hessian matrix of Q). Several researchers have proposed decomposition methods to solve this optimization problem [16] [203] [136] [119] [82] [90] [144]. Vapnik et al. describe a "chunking" method [16] using the fact that the solution of a QP problem is the same if we remove the rows and columns of the matrix Q that correspond to zero Lagrange multipliers. Thus, a large QP problem can be decomposed into a series of smaller QP subproblems in which all of the nonzero Lagrange multipliers are identified and all the zero Lagrange multipliers are discarded. After all the nonzero Lagrange multipliers in Q have been identified, the last step then solves the remaining QP problem. Osuna et al. propose a novel decomposition algorithm for solving the SVM QP problem [136]. As a variation of the active set methods, a large QP problem is decomposed into a series of subproblems by maintaining a small working set. The algorithm works by moving the samples that violate the Karush-Kühn-Tucker condition to the working set in each iteration, and solve the subproblems. Since the working set is usually small, this method does not have memory problems. However, a numerical QP solver is required. Joachims improves Osuna's methods with a strategy to select good working sets [82]. Platt develops Sequential Minimal Optimization (SMO) which further improves Osuna's method by making all the working sets of size 2 with a set of heuristics [144]. One feature of SMO is that all the subproblems are solved analytically.

An analogous problem to SVM has been investigated in the Statistical Mechanics literature, which has resulted in several perceptron-alike algorithms. These algorithms, e.g., Adatron [7], aim to find maximal margin hyperplanes in the input space. By introducing kernels into Adatron, Friess

et al. propose Kernel-Adatron which is able to maximize the margin in the feature space [55]. Ruján describes a stochastic approximation method to to estimate the optimal Bayes classifier for linearly separable problems [164] and later extends to support vector machines using kernels [165]. Using a projection method different fromwhat has been used in conventional SVMs, Freund and Shcapire develop an algorithm to find maximal margin perceptron in a high dimensional feature space [53]. Compared to SVMs, this method is much simpler to implement (i.e., no QP optimization) and more efficient in terms of computational complexity. However the performance is close to, but not as good as, the performance of SVM on the same problems.

Recently Keerthi et al. treat SVM classification in terms of finding the nearest points between pairs of convex polytopes in the feature space, and solve them with an iterative nearest point algorithm. Although this approach does not require numerical QP library, one drawback is that it needs to consider $\frac{m(m-1)}{2}$ pairs of points for a problem with m points.

7.3 Extracting Guard Vectors for Support Vector Machines

In this section we first describe a method to extract the guard vectors by solving a set of linear programming problems. These guard vectors are then used to construct hyperplane for linear SVMs. For nonlinear SVMs, a projection method is described to project vectors onto a higher dimensional space in which a linear hyperplane is constructed.

7.3.1 Determining Guard Vectors Using Linear Programming

A point in a plane has two parameters: its x-coordinate and its y-coordinate. A line in a plane has also two parameters: its slope and its intersection with the y-axis. The duality transform in computational geometry specifies that every point in a plane has a unique dual hyperplane representation [148]. In other words, there exists a one-to-one mapping between points and lines such that certain properties of points translate to certain other properties for the set of lines. For instance, the mapping is incidence and order preserving. The dual of a point $p = (p_x, p_y)$ in a plane is a line defined by $y = p_x x - p_y$. Figure 7.1 shows each point in the primal plane corresponds to a line in the dual plane. Also note that three points on a line becomes three lines through a point.

For each point p_i, we can use its class label to determine whether there exists a line l_s through

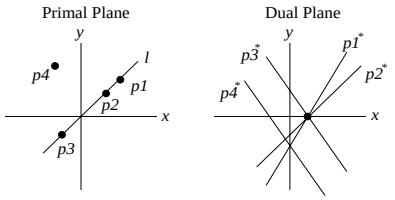


Figure 7.1: Dual representation of a point. Each point in the primal plane corresponds to a line in the dual plane.

p_i in the primal plane such that all the points of the same label are on the same side of l_s. If all the points of the same label fall on the same side of l_s, it means l_s can linearly separate all the points. We call such point p_i a guard vector. Since each point on a plane corresponds to a line, we can find all the dual line representations for all the points. Therefore, the problem of determining the existence of l_s is equivalent to solving a linear program in the dual space where the constraints are the dual lines of the points in the primal plane. If there exists a feasible point to the LP in the dual plane, it means there exists a line in the primal plane that linearly separates all the points.

Figure 7.2 shows three points with their duals (i.e., lines) in a plane in which points p_1 and p_2 belong to class '+' while point p_3 belongs class '-'. Consider point p_1 first, Figure 7.2 (a) shows one LP that we use the dual of point 1 ,i.e., line l_1, and specifies that all the points belonging to class '+' should be on the right hand side of or on l_1 and the points belonging to class '-' should be on the left hand side of l_1. The corresponding LP for point p_1 is

$$p_{11}x_1 - x_2 = p_{12} \quad (l_1)$$
$$p_{21}x_1 - x_2 \geq p_{22} \quad (l_2)$$
$$p_{31}x_1 - x_2 \leq p_{32} \quad (l_3)$$

Note the equality and inequality constraints are set according to the class labels of the point considered in this LP and the other points. This means that any solution $p^* = (x_1, x_2)$ must be on l_1 and p^* must satisfy the constraints in l_2 and l_3. In other words, if there exists a solution for this LP, l_2 should be on the right hand side of l_1 and l_3 should be on the left hand side of l_1. Clearly, there is no feasible solution to satisfy all the constraints since p^* does not satisfy the constraint in l_3. Figure 7.2 (b) shows another possible formulation for l_1. The constraint of l_2 cannot be satisfied and thus there is no feasible solution for this formulation. Since there is no feasible solution for

either formulation of l_1, p_1 is not a guard vector. Similarly for p_2, Figure 7.2 (c) shows one case that the one LP that has no feasible solution. However, Figure 7.2 (d) shows one LP of p_2 that has feasible solution because l_1 is on the right left hand side of l_2 and l_3 is on the right hand side of l_2. In fact, the shaded area in Figure 7.2 shows the all the feasible solutions for this LP. Consequently, p_2 is a guard vector. This example illustrates the use of LP in which a feasible solution means the intersection of the half planes specified by the constraints is nonempty. It also shows that we are only interested in the feasibility of a LP rather than its optimal value.

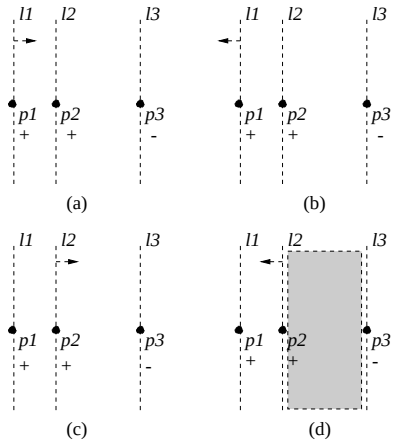


Figure 7.2: Using linear programming to extract guard vectors. Each point and its dual are shown in the figure. Points p_1 and p_2 belong to class '+' while point p_3 belongs to class '-'.

Formally, for a set of m points with q positive examples and $m - q$ negative examples, we can formulate a LP problem for point p_1:

$$
\begin{array}{lllllll}
\text{Maximize } x_1 & & & & & & \\
\text{Subject to} & p_{11}x_1 & - & x_2 & = & p_{12} \\
& p_{21}x_1 & - & x_2 & \geq & p_{22} \\
& & & & \cdots & \\
& p_{q1}x_1 & - & x_2 & \geq & p_{q2} \\
& p_{(q+1)1}x_1 & - & x_2 & \leq & p_{(q+1)2} \\
& & & & \cdots & \\
& p_{m1}x_1 & - & x_2 & \leq & p_{m2}
\end{array}
$$

where, without loss of generality, we assume that p_i, $1 \leq i \leq q$ are positive examples and the others

are negative examples. The constraints in the LP specify the correct class labels for each point and the LP problem itself means that we want to find a feasible solution to satisfy all the constraints. Figure 7.3 gives a geometric interpretation of an LP problem. If there is a feasible solution in the dual plane, it means that we can find a hyperplane through p_1 in the primal plane such that all the points of the same class fall on the same side of the hyperplane (i.e., have the same class label). In other words, the intersection of the half planes specified by all the constraints is not empty as shown in Figure 7.3 (a). Since we do not have prior knowledge about whether the points of the same class are above or below the hyperplane, we need to change the inequality sign to check the other possibility. A point is a guard vector if there exists one feasible solution to the corresponding LP. If there is no feasible solution to the LP, it means that there is no existing hyperplane such that all the points are separated correctly. This means the intersection of all the half planes specified by the constraints is empty as shown in Figure 7.3 (b). Note that the exact form of the objective function is immaterial since we are interested in whether there is a solution to the constraints or not, which means the optimization problem can be solved more efficiently if we choose a simple form.

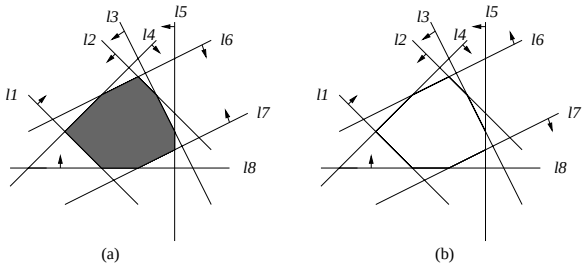


Figure 7.3: Geometric interpretation of a linear programming problem. (a) shows one case where a feasible solution exists. (b) shows one case where no feasible solution exists.

Figure 7.4 shows a set of 50 points belonging to two classes: the points labeled with '+' belong to class 1 and the others labeled with '*' belong to class 2. The guard vectors are labeled with circles and the optimal separating hyperplane found by a linear SVM is the dash-dot line. For point p_{17}, the corresponding LP does not have a feasible solution and therefore it is not a guard vector. In other words, it means that there exists on line such that all the points can be correctly separated. Similarly point p_{45} is not a guard vector. For each guard vector, there exists one line such that all the points can separated correctly. A close study shows that guard vector 1 and 4

are support vectors for the optimal separating hyperplane found by a linear SVM. Also note that guard vectors form a superset of support vectors.

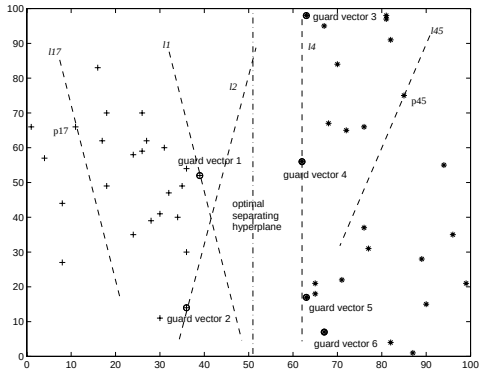


Figure 7.4: Geometric interpretations of guard vectors and support vectors (guard vectors form a superset of support vectors): guard vector 1 and 4 are the support vectors for the optimal separating hyperplane found by the SVM for this set of points.

Duality applies to high dimensional point sets as well. For a point in n-dimensional space, $p_i = (p_{i1}, p_{i2}, \ldots, p_{in})$, its dual p^* is the hyperplane $x_d = p_{i1}x_1 + p_{i2}x_2 + \ldots + p_{in-1}x_{n-1} - p_{in}$. Similar to the 2D case, we can use solve a linear programming problem to determine whether a point is a guard vector or not:

$$Ax \leq b$$

In matrix form, we have

$$\begin{bmatrix} -p_{11} & -p_{12} & \cdots & +1 \\ -p_{21} & -p_{22} & \cdots & +1 \\ \vdots & \vdots & \ddots & \\ p_{m1} & p_{m2} & \ldots & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} -p_{1n} \\ -p_{2n} \\ \vdots \\ p_{mn} \end{bmatrix}$$

where m is the number of points and the sign of each element depends on its class label and the class label of the point that the LP corresponds to.

We summarize the abovementioned algorithm as follows:

Algorithm: Finding the guard vectors

1. Transform each point to its dual hyperplane representation with inequality '$\geq$'. Initialize i to 1.

2. Consider point p_i with class label l_i, first set the equality sign for dual hyperplane for point p_i. Then set the inequality signs of all dual hyperplanes of the points having the same class label to '$\geq$', and set the inequality signs of all the other hyperplanes of the points having the opposite class label to '$\leq$'.

3. Solve the current LP problem. If there is a feasible solution then it is a guard vector, else reverse all the inequality signs. If there is still no feasible solution to the new LP, then point p_i is not a guard vector

4. Increase i by 1 and go to step 2 until all the points have been checked.

7.3.2 Linear SVM

Given a set of samples $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_l, y_m)$ where $\mathbf{x}_i$ ($\mathbf{x}_i \in R^n$) is the input vector of n dimension and y_i is its label ($y_i \in \{-1, 1\}$) for a recognition problem, SVM aims to find the optimal hyperplane that leaves the largest possible fraction of data points of the same class on the same side while maximizing the distance of either class from the hyperplane (margin). Vapnik [203] shows that maximizing the margin distance is equivalent to minimizing the VC dimension in constructing an optimal hyperplane. The optimal hyperplane is in the form

$$f(\mathbf{x}) = \sum_{i=1}^{m} \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{m} y_i \alpha_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b$$

where $k(\mathbf{x}, \mathbf{x}_i) = \mathbf{x} \cdot \mathbf{x}_i$ for linear SVMs. The sign of $f(\mathbf{x})$ determines the class label of $\mathbf{x}$.

Since the optimal hyperplane can be constructed *exactly* using only the support vectors, and the guard vectors form a superset of SVs. We can use the proposed algorithm to extract all the guard vectors and then find the support vectors and corresponding nonzero α_i by solving a small QP to construct the optimal hyperplane.

7.3.3 Nonlinear SVM

For the patterns that cannot be lineally separated, we use a nonlinear function similar to [53] to project training vectors from input space to a high dimensional space in which we construct a linear hyperplane. For better generalization performance, we may also want to project training vectors from input space to a higher dimensional space. One distinct feature of this projection function is that the projected vectors are guaranteed to be linearly separated in the projected space.

Let $\mathbf{X}$ be a Hilbert space and for any fixed nonnegative Δ, we define a projection of $\mathbf{X}$ onto another higher dimensional Hilbert space $\mathbf{X}'$ as

$$\tau_\Delta : \mathbf{x}_i \in \mathbf{X} \mapsto \mathbf{x}_i' = (\mathbf{x}_i, \Delta\delta_{\mathbf{x}_i}) \in \mathbf{X}'$$

where $\delta_{\mathbf{x}_i} \in R^m$ is defined by

$$\delta_{\mathbf{x}_i}(j) = \begin{cases} 1; & \text{if } \ j = i; 1 \leq j \leq m \\ 0; & \text{otherwise.} \end{cases}$$

Given a set of samples $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_m, y_m)$ where $\mathbf{x}_i$ ($\mathbf{x}_i \in R^n$) is the input vector of n dimension and y_i is its label ($y_i \in \{-1, 1\}$), the projection extends the input space R^n to R^{n+m} by adding m new dimensions, one for each example. Let $\mathbf{x}_i' \in R^{n+m}$ denote the extension of the point $\mathbf{x}_i$, we set the first n coordinates of $\mathbf{x}_i'$ equal to $\mathbf{x}_i$ and the $(n+i)$-th coordinate to Δ. The rest of the coordinates of $\mathbf{x}_i'$ are set to zero.

For a linear classifier $\mathbf{w}$ on $\mathbf{X}$ and threshold $b \in R^n$, we define deviation d_i for each example as

$$d_i((\mathbf{x}_i, y_i), (\mathbf{w}, b), \gamma) = \max\{0, \gamma - y_i((\mathbf{w} \cdot \mathbf{x}_i) - b)\}$$

where d_i is the amount by which $\mathbf{w}$ fails to reach the margin at point $(\mathbf{x}_i, y_i)$ or 0 if its margin is larger than γ, i.e.

$$d_i((\mathbf{x}_i, y_i), (\mathbf{w}, b), \gamma) \geq \gamma - y_i((\mathbf{w} \cdot \mathbf{x}_i) - b)$$

Only those points whose margins are less than γ have $d_i > 0$ cause *errors* in classification.

Similar to the projection for $\mathbf{x}_i$, we project the normal vector $\mathbf{w}$ of a hyperplane $\in R^n$ to $\mathbf{w}^{'} \in R^{n+m}$ by the following function:

$$\mathbf{w}^{'} = (\mathbf{w}, \frac{1}{\Delta}\sum_{i=1}^{m} d((\mathbf{x}_i, y_i), (\mathbf{w}, b), \gamma) y_i \delta_{\mathbf{x}_i})$$

Claim: All the points $\mathbf{x}^{'}$ in the projected space $\mathbf{X}^{'}$ are linearly separable.

Proof: For a point $(\mathbf{x}_i, y_i)$ and a hyperplane $(\mathbf{w},\ b)$, we have the following in the projected space $X^{'}$,

$$\begin{aligned}
& y_i^{'}((\mathbf{w}^{'} \cdot \mathbf{x}_i^{'}) - b) \\
&= y_i^{'}((\mathbf{w}, \mathbf{x}^{'}) - b) + y_i^{'}(\textstyle\sum_{i=1}^{m} d((\mathbf{x}_i, y_i), (\mathbf{w}, b), \gamma) y_i \delta_{\mathbf{x}_i} \cdot \delta_{\mathbf{x}_i^{'}}) \\
&\geq \gamma - d_i((\mathbf{x}_i^{'}, y_i^{'}), \mathbf{w}, \gamma) + d((\mathbf{x}_i^{'}, y_i^{'}, \mathbf{w}, \gamma) = \gamma
\end{aligned}$$

In other words, all the points in Hilbert space $\mathbf{X}^{'}$ have margins that are large than γ. Therefore, all the points can be linearly separated.

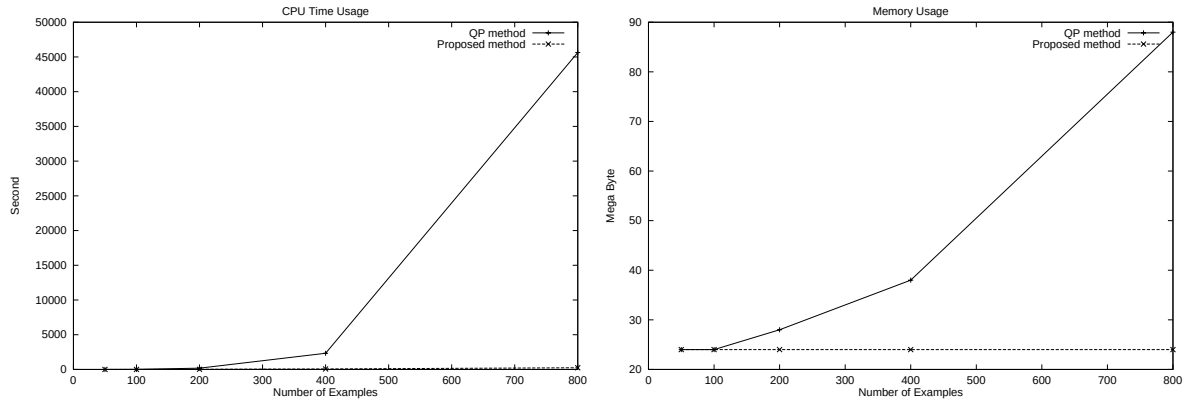
7.4 Experiments

In this section, the proposed algorithm is tested against SVMs with QP numerical library on synthetic and benchmark data. All implementations are written in MATLAB and the CPU times of all the experiments are measured on an unloaded Sun Ultra 10. The objective of these experiments is to benchmark the CPU time and memory usage of the proposed method and the conventional QP approach in SVMs, rather than the recognition performance. In other words, we do not tune the parameters in SVMs to get the optimal performance (e.g., the parameter C is set to 1000 for all experiments). However, both methods do produce the same optimal separating hyperplane in each experiment.

7.4.1 Synthetic Data

Each synthetic data set is randomly generated from a n-dimensional hypercube. For each data set, $m/2$ positive examples are generated from a hypercube of $[0, 40]^n$ and $m/2$ negative examples are

generated from a hypercube of $[60, 100]^n$. Figure 7.5 shows the CPU time and memory requirements on a series of synthetic data setsin 2 dimensions. The experimental results show that the proposed method outperforms the QP implementation for SVM since it requires much less CPU time and memory. A close examination also shows that the number of guard vectors found by the proposed method is at most twice the number of support vectors.



(a) (b)

Figure 7.5: Benchmark results of the proposed algorithm against a linear SVM in terms of CPU time and memory requirements using the first 2-dimensional data set shown in Table 7.1.

Table 7.1 shows more experimental results in which points are generated from hypercubes of higher dimensions. Note the reported CPU time of the proposed method in each entry includes the time to extract guard vectors and the time to train an SVM using the conventional QP method. The results show that the proposed algorithm performs well when the ratio between the number of points and the dimensionality (i.e., m/n) is high. Meanwhile, the ratio between the number guard vectors and the number of support vectors depends both on the ratios of m/n and the structure of the data set itself.

7.4.2 Real-World Data

The first data set we use is from the Wisconsin Breast Cancer data set [120]. Each of the 699 data elements has 9 attributes, with class label indicating whether it represents a benign or malignant example. The second data set consists of 500 20×20 face images and 1000 20×20 nonface face

Table 7.1: Experimental results on synthetic data: Each data set has m points that are randomly scattered in an n-dimensional hypercube.

Dimension (n): 2								
	Proposed method			QP method			Ratio	
# pt (m)	cpu (sec)	mem (MB)	# gv (g)	cpu (sec)	mem (MB)	# sv(s)	g/s	m/n
50	1.96	24	3	1.63	24	2	1.5	25.0
100	5.61	24	4	8.50	24	3	1.3	50.0
200	18.4	24	4	187.2	28	3	1.3	100.0
400	64.57	24	8	2326.69	38	5	1.6	200.0
800	243.19	24	11	45633.23	88	6	1.9	400.0
Dimension (n): 3								
	Proposed method			QP method			Ratio	
# pt (m)	cpu (sec)	mem (MB)	# gv (g)	cpu (sec)	mem (MB)	# sv(s)	g/s	m/n
50	2.23	24	16	1.91	24	3	5.3	16.6
100	6.31	24	29	11.9	25	6	4.8	33.3
200	20.73	24	35	307.85	28	14	2.5	66.6
400	70.64	24	46	3717.81	39	23	2.0	133.3
Dimension (n): 5								
	Proposed method			QP method			Ratio	
# pt (m)	cpu (sec)	mem (MB)	# gv (g)	cpu (sec)	mem (MB)	# sv(s)	g/s	m/n
50	2.6	24	40	1.84	24	2	12.0	10.0
100	7.2	24	61	19.18	24	4	15.3	20.0
200	23.41	24	130	178.51	28	12	10.8	40.0
400	78.10	24	181	1885.75	45	10	18.1	80.0

Table 7.2: Experimental results on benchmark data sets: The first one is the Wisconsin Breast Cancer data set in which each of the 699 points has 9 features. The second data set consists of 500 20×20 face images and 1000 20×20 nonface images for face detection.

	Proposed method			QP method			Ratio	
# pt (m)	cpu (sec)	mem (MB)	# gv (g)	cpu (sec)	mem (MB)	# sv(s)	g/s	m/n
699	99.02	28	317	3572.91	68	52	6.1	77.7
1500	39086.21	38	712	36506400.81	168	347	4.2	3.8

images, to be used for face detection testssimilar tothose reported in [136]. Both data sets are trained using a linear SVM for benchmarking. Figure 7.6 shows some of the face images in the data set.

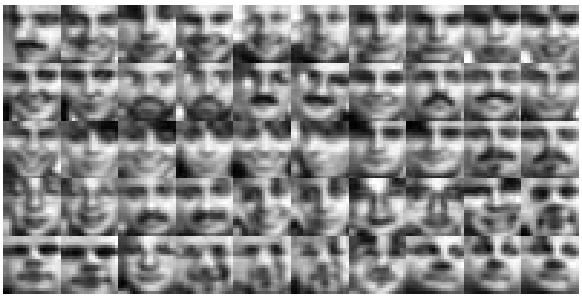


Figure 7.6: Some 20×20 face images used in the experiment. Each image is converted to a 400-dimensional vector.

Table 7.2 shows the CPU time and memory usage of both methods. For both cases, the proposed algorithm is more than 30 times faster than the conventional QP approach. Furthermore, the memory requirement of the proposed method is one fourth that of the QP method in a large scale experiment (i.e. the data set for face detection). The results also show that the set of guard vectors is a small superset (i.e., roughly 20 times) of support vectors. Note that both the proposed and conventional methods generate the same optimal hyperplane in all the experiments.

7.5 Discussion and Conclusion

The optimal decision surface of an SVM is constructed from its support vectors which are conventionally determined by solving a quadratic programming problem. However, solving a large optimization problem is challenging since it is computationally intensive and the memory requirement grows with the square of the number of training vectors. In this chapter, we have proposed a geometric method to extract guard vectors, a small superset of support vectors, to construct the optimal decision surface. Specifically, the superset of support vectorsis found by solving a set of linear programming problems. Experimental results on synthetic and real data sets show that the proposed method is more efficient than conventional methods using QPs and requires much less memory.

Future work will focus on theoretical analysis of the number of guard vector versus the number

of support vectors. Although the numbers of guard vectors and support vectors depend on how the data points scatter in the input and feature space, we believe the relationship between the number of guard and support vectors can be characterized with some assumptions such as general position of points. More experiments will also be conducted to benchmark the proposed algorithm against other training methods for SVMs.

Chapter 8

Conclusion and Future Work

.

In this thesis, various aspects of research on intelligent human computer interaction are discussed in the context of computer vision and machine learning. In this chapter, we summarize the contributions of this work and sketch future research directions.

8.1 Conclusion

In Chapters 3 and 4, we have first described a method to recognize motion patters using 2-D motion trajectories from image sequences. We have demonstrated the advantages of this method for recognizing hand gestures of American Sign Language. Experimental results on a set of 40 ASL gestures show that motion patterns in hand gestures can be extracted [217] and recognized with high accuracy using motion trajectories [218] [220].

A problem related to hand gesture recognition is face detection. We have presented a comprehensive survey of face detection and related problems in Chapter 2. We have developed methods to detect faces in grayscale and color images in Chapter 5. Compared with most state-of-the-art face detection methods, our methods [222] [226] [223] perform equally well in terms of detection rate, but have much fewer false detects.

Many pattern recognition problems in computer vision can be posed as learning problems. We have presented an account of learnability for object recognition within the PAC framework in Chapter 6. The evaluation we provide for this framework relies on the SNoW learning architecture that is used in a large scale object recognition experiment [158]. Experimental results show that

the SNoW-based method outperforms the SVM-based system in terms of recognition rate and the computational cost involved in learning [224] [225]. Most importantly, SNoW's performance degrades gracefully when the training data contains fewer views.

Training a SVM for a large-scale problem is challenging since it is computationally intensive and the memory requirement grows with square of the number of training vectors. In Chapter 7, we have proposed a geometric method to extract a small superset of support vectors, which we call guard vectors, to construct the optimal decision surface. Experimental results on synthetic and real data sets show that the proposed method is more efficient than conventional methods and requires much less memory [221].

8.2 Future Work

ASL experts have been using the shape, location and movement of hands for indexing and categorization [208]. For example, a sign is indexed to have "opening" shape at beginning and "move upward" "from the lower torso", and then "make a circle," and then finish with a "closing" shape at "upper forehead". In other words, ASL experts use their expertise to define a set of formation features such as hand shapes and movements that frequently appear in signs. They then use these features to describe a sign with respect to certain human parts.

In Chapter 3, we have demonstrated that our method can recognize hand gestures using the motion trajectories associated with the hands of a signer. One extension is to develop a system that can automatically index ASL signs with the same concept (i.e., shape, location and movement information) by ASL experts. First, hand shape information can be extracted from the detected hand regions in video sequences discussed in Chapter 3. Feature vectors can be computed, using geometric moments, to describe the configurations of the hands in different gestures. These feature vectors can then be clustered using Kohonen's Self-Organizing Map algorithm. Consequently, we can find salient shapes from the gesture database. Second, similar to what has been done in our gesture recognition method [217] [218] [220], the detected human head is used as the origin reference coordinate to describe the location of hand shapes. Third, we can segment the extracted motion trajectories into "movement segments" using a clustering algorithm (e.g., Kohonen's Self-Organizing Map). Note that a motion trajectory in our method is a pixel flow in the hand region

across frames. In other words, we can estimate the velocity and acceleration of a pixel across image frames. The movement segments are identified by searching the pixels where the velocity and direction of the pixel flow changes significantly (i.e., acceleration). In addition, the pixels where the shape change significantly are used to segment the trajectories. These segments are then normalized (to have similar length) and clustered (e.g., using Kohonen's Self-Organizing Map) into classes. Consequently, we can index gestures using the extracted shape, location and movement information. Finally, a similar video can be retrieved based on similarity between motion contents of the query, indexed by the abovementioned process, and the video database.

The methods developed in Chapter 5 detect only upright and frontal faces. It is of great interest to extend the current work to detect faces in arbitrary pose and orientation. One simple but time-consuming scheme is to use the routing network proposed by Rowley [161]. Meanwhile, it is important to speed up the methods in Chapter 5 since the detection process searches exhaustively in a test image. It is important to develop a method such that "regions of interest" can be identified promptly. Then we only need to apply the developed to search faces in these regions.

The work presented in Chapter 6 gives a PAC-based learning account for object recognition and empirical results on a large-scale experiment. However, the experiments were performed in an ideal environment. We believe this work, which uses conjunctions of edge information to represent objects, can be extended to recognize objects in arbitrary background and view points if we have enough examples of the objects. Since there may exist a large number of edges in an image, it is of interest to develop efficient methods that can compute conjunctions of edge information with low time complexity.

In Chapter 7, we have presented a geometric method to extract guard vectors to construct an optimal decision surface. Future work will focus on theoretical analysis of the number of guard vectors versus the number of support vectors. Although the number of guard vectors and support vectors depend on how the data points scatter in the input and feature space, we believe their relationship can be characterized with some assumptions such as general position of points. More experiments need to be conducted to benchmark the proposed algorithm against other training methods for SVMs.

References

[1] T. Agui, Y. Kokubo, H. Nagashashi, and T. Nagao. Extraction of face recognition from monochromatic photographs using neural networks. In *Proceedings of the Second International Conference on Automation, Robotics and Computer Vision*, volume 1, pages CV–18.8.1–CV–18.8.5, 1992.

[2] N. Ahuja. A transform for multiscale image segmentation by integrated edge and region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):1211–1235, 1996.

[3] M. Aitkin, D. Anderson, and J. Hinde. Statistical modeling of data on teaching styles. *Journal Royal Statistics Society*, 144:419–461, 1981.

[4] Y. Amit, D. Geman, and B. Jedynak. Efficient focusing and face detection. In H. Wechsler, P. J. Phillips, V. Bruce, F. Fogelman-Soulie, and T. S. Huang, editors, *Face Recognition: From Theory to Applications*, volume 163 of *NATO ASI Series F, Computer and Systems Sciences*, pages 124–156. Springer, 1998.

[5] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1300–1305, 1997.

[6] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. John Wiley, New York, 1984.

[7] J. K. Anlauf and M. Biehl. The adatron: An adaptive perceptron algorithm. *Europhysics Letters*, 10(7):687–692, 1989.

[8] M. F. Augusteijn and T. L. Skujca. Identification of human faces through texture-based

feature recognition and neural network technology. In *Proceedings of IEEE Conference on Neural Networks*, pages 392–398, 1993.

[9] M. S. Bartlett, P. A. Viola, T. J. Sejnowski, B. Golomb, J. Larsen, J. C. Hager, and P. Ekman. Classifying facial action. In *Advances in Neural Information Processing Systems 8*, pages 823–829. MIT Press, 1996.

[10] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[11] O. Bernier, M. Collobert, R. Feraud, V. Lemarie, J. E. Viallet, and D. Collobert. MULTRAK: A system for automatic multiperson localization and tracking in real-time. In *Proceedings of International Conference on Image Processing*, pages 136–140, 1998.

[12] M. J. Black and A. D. Jepson. A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gesture and expressions. In *Proceedings of the Fifth European Conference on Computer Vision*, pages 909–924, 1998.

[13] A. Blum. Learning boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386, 1992.

[14] F. L. Bookstein. A statistical method for biological shape comparison. *Journal of Theoretical Biology*, 107:475–520, 1984.

[15] F. L. Bookstein. Size and shape spaces for landmark data in two dimensions. *Statistical Science*, 1(2):181–242, 1986.

[16] B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 1992.

[17] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and regression trees.* Wadsworth, 1984.

[18] G. Burel and D. Carel. Detection and localization of faces on digital images. *Pattern Recognition Letters*, 15(10):963–967, 1994.

[19] M. C. Burl, T. K. Leung, and P. Perona. Face localization via shape statistics. In *Proceedings of the First International Workshop on Automatic Face and Gesture Recognition*, pages 154–159, 1995.

[20] J. Cai, A. Goshtasby, and C. Yu. Detecting human faces in color images. In *Proceedings of the 1998 International Workshop on Multi-Media Database Management Systems*, pages 124–131, 1998.

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

[22] A. Carleson, C. Cumby, J. Rosen, and D. Roth. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department, 1999.

[23] D. Chai and K. N. Ngan. Locating facial region of a head-and-shoulders color image. In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 124–129, 1998.

[24] R. Chellappa, C. L. Wilson, and S. Sirohey. Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, 83(5):705–740, 1995.

[25] Q. Chen, H. Wu, and M. Yachida. Face detection by fuzzy matching. In *Proceedings of the Fifth IEEE International Conference on Computer Vision*, pages 591–596, 1995.

[26] D. Chetverikov and A. Lerch. Multiresolution face detection. In *Theoretical Foundations of Computer Vision*, volume 69 of *Mathematical Research*, pages 131–140. Akademie Verlg, 1993.

[27] R. Cipolla and A. Blake. The dynamic analysis of apparent contours. In *Proceedings of the Third IEEE International Conference on Computer Vision*, pages 616–623, 1990.

[28] M. Collobert, R. Feraud, G. L. Tourneur, O. Bernier, J. E. Viallet, Y. Mahieux, and D. Collobert. LISTEN: A system for locating and tracking individual speakers. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 283–288, 1996.

[29] A. J. Colmenarez and T. S. Huang. Maximum likelihood face detection. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 307–311, 1996.

[30] A. J. Colmenarez and T. S. Huang. Face detection with information-based maximum discrimination. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 782–787, 1997.

[31] T. F. Cootes and C. J. Taylor. Locating faces using statistical feature detectors. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 204–209, 1996.

[32] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20, 1995.

[33] T. Cover and J. Thomas. *Elements of Information Theory.* Wiley Interscience, 1991.

[34] I. Craw, H. Ellis, and J. Lishman. Automatic extraction of face features. *Pattern Recognition Letters*, 5:183–187, 1987.

[35] I. Craw, D. Tock, and A. Bennett. Finding face features. In *Proceedings of European Conference on Computer Vision*, pages 92–96, 1992.

[36] J. L. Crowley and J. M. Bedrune. Integration and control of reactive visual processes. In *Proceedings of the Third European Conference on Computer Vision*, volume 2, pages 47–58, 1994.

[37] J. L. Crowley and F. Berard. Multi-modal tracking of faces for video communications. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 640–645, 1997.

[38] Y. Dai and Y. Nakano. Extraction for facial images from complex background using color information and SGLD matrices. In *Proceedings of the First International Workshop on Automatic Face and Gesture Recognition*, pages 238–242, 1995.

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

[40] I. L. Dryden and K. V. Mardia. General shape distribution in a plane. *Advanced Applied Probability*, 23:259–276, 1991.

[41] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley, New York, 1973.

[42] N. Duta and A. K. Jain. Learning the human face concept from black and white pictures. In *International Conference on Pattern Recognition*, pages 1365–1367, 1998.

[43] S. Edelman. On learning to recognize 3-D objects from examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):833–837, 1993.

[44] G. J. Edwards, C.J.Taylor, and T. Cootes. Learning to identify and track faces in image sequences. In *Proceedings of British Machine Vision Conference*, pages 130–139, 1997.

[45] G. J. Edwards, C.J.Taylor, and T. Cootes. Learning to identify and track faces in image sequences. In *Proceedings of the Sixth IEEE International Conference on Computer Vision*, pages 317–322, 1998.

[46] B. Efron. Bootstrap methods: another look at the jackknife. *Ann. Statist.*, 7:1–26, 1979.

[47] I. A. Essa and A. Pentland. Facial expression recognition using a dynamic model and motion energy. In *Proceedings of the Fifth IEEE International Conference on Computer Vision*, pages 360–367, 1995.

[48] S. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2*, pages 524–532, 1990.

[49] R. Feraud. PCA, neural networks and estimation for face detection. In H. Wechsler, P. J. Phillips, V. Bruce, F. Fogelman-Soulie, and T. S. Huang, editors, *Face Recognition: From Theory to Applications*, volume 163 of *NATO ASI Series F, Computer and Systems Sciences*, pages 424–432. Springer, 1998.

[50] R. Feraud and O. Bernier. Ensemble and modular approaches for face detection: A comparison. In *Advances in Neural Information Processing Systems 10.* MIT Press, 1998.

[51] F. Fleuret and D. Geman. Graded learning for object detection. In *Proceedings of the IEEE workshop on Statistical and Computational Theories of Visions*, 1999.

[52] D. Forsyth. A novel approach to color constancy. *International Journal of Computer Vision*, 5(1):5–36, 1990.

[53] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 1999. To appear.

[54] B. J. Frey, A. Colmenarez, and T. S. Huang. Mixtures of local subspaces for face recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 32–37, 1998.

[55] T. Friess, N. Cristianini, and C. Campbell. The kernel-adatron: a fast and simple learning procedure for support vector machines. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 188–196, 1998.

[56] F. Fukunaga and W. Koontz. Applications of the Karhunen-Loève expansion to feature selection and ordering. *IEEE Transactions on Computers*, 19(5):311–318, 1970.

[57] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic, New York, 1972.

[58] Z. Ghahramani and G. E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto, 1996. Available at ftp://ftp.cs.toronto.edu/pub/zoubin/tr-96-1.ps.gz.

[59] R. C. Gonzalez and P. A. Wintz. *Digital Image Processing.* Addison Wesley, Reading, 1987.

[60] V. Govindaraju. Locating human faces in photographs. *International Journal of Computer Vision*, 19(2):129–146, 1996.

[61] V. Govindaraju, D. B. Sher, R. K. Srihari, and S. N. Srihari. Locating human faces in newspaper photographs. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 549–554, 1989.

[62] V. Govindaraju, S. N. Srihari, and D. B. Sher. A computational model for face location. In *Proceedings of the Third IEEE International Conference on Computer Vision*, pages 718–721, 1990.

[63] H. P. Graf, T. Chen, E. Petajan, and E. Cosatto. Locating faces and facial parts. In *Proceedings of the First International Workshop on Automatic Face and Gesture Recognition*, pages 41–46, 1995.

[64] H. P. Graf, E. Cosatto, D. Gibbon, M. Kocheisen, and E. Petajan. Multimodal system for locating heads and faces. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 88–93, 1996.

[65] D. B. Graham and N. M. Allinson. Characterizing virtual eigensignatures for general purpose face recognition. In H. Wechsler, P. J. Phillips, V. Bruce, F. Fogelman-Soulie, and T. S. Huang, editors, *Face Recognition: From Theory to Applications*, volume 163 of *NATO ASI Series F, Computer and Systems Sciences*, pages 446–456. Springer, 1998.

[66] P. Hallinan. *A Deformable Model for Face Recognition Under Arbitrary Lighting Conditions*. PhD thesis, Harvard University, 1995.

[67] C.-C. Han, H.-Y. M. Liao, K.-C. Yu, and L.-H. Chen. Fast face detection via morphology-based pre-processing. In *Proceedings of the Ninth International Conference on Image Analysis and Processing*, pages 469–476, 1998.

[68] R. M. Haralick, K. Shanmugam, and I. Dinstein. Texture features for image classification. *IEEE Transactions on System, Man and Cybernetics*, 3(6):610–621, 1973.

[69] T. Hastie and W. Stuetzle. Principal curves. *Journal of American Statistical Association*, 84(406):502–516, 1989.

[70] D. Haussler, M. Kearns, N. Littlestone, and M. K. Warmuth. Equivalence of models for polynomial learnability. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 42–55, Cambridge, MA, 1988. Morgan-Kaufmann.

[71] D. M. Hawkins. A new test for multivariate normality and homoscedasticity. *Technometrics*, 23:105–110, 1981.

[72] G. E. Hinton, P. Dayan, and M. Revow. Modeling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks*, 8(1):65–74, 1997.

[73] A. C. A. Hope. A simplified Monte Carlo significance test procedure. *Journal of Royal Statistics Society*, 30:582–598, 1968.

[74] B. Horn. *Robot vision*. McGraw-Hill, 1986.

[75] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 498–520, 1933.

[76] K. Hotta, T. Kurita, and T. Mishima. Scale invariant face detection method using higher-order local autocorrelation features extracted from log-polar image. In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 70–75, 1998.

[77] J. Huang, S. Gutta, and H. Wechsler. Detection of human faces using decision trees. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 248–252, 1996.

[78] D. Hutenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:850–863, 1993.

[79] M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[80] T. S. Jebara and A. Pentland. Parameterized structure from motion fro 3D adaptive feedback tracking of faces. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 144–150, 1997.

[81] T. S. Jebara, K. Russell, and A. Pentland. Mixtures of eigenfeatures for real-time structure from texture. In *Proceedings of the Sixth IEEE International Conference on Computer Vision*, pages 128–135, 1998.

[82] T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, chapter 11, pages 169–184. MIT Press, 1998.

[83] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 73(2):201–211, 1973.

[84] I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, New York, 1986.

[85] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 274–280, 1999.

[86] P. Juell and R. Marsh. A hierarchical neural network for human face detection. *Pattern Recognition*, 29(5):781–787, 1996.

[87] T. Kanade. *Picture processing by computer complex and recognition of human faces.* PhD thesis, Kyoto University, 1973.

[88] K. Karhunen. Uber lineare methoden in der wahrscheinlichkeitsrechnung. *Annales Academiae Sciientiarum Fennicae, Series AI: Mathematica-Physica*, 37:3–79, 1946. (Translated: RAND Corp., Santa Monica, CA, Report T-131, August, 1960).

[89] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proceedings of the First IEEE International Conference on Computer Vision*, pages 259–269, 1987.

[90] L. Kaufman. Solving the quadratic programming problem arising in support vector classification. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, chapter 10, pages 147–167. MIT Press, 1998.

[91] R. Kauth, A. Pentland, and G. Thomas. Blob: An unsupervised clustering approach to spatial preprocessing of MSS imagery. In *Proceedings of the Eleventh International Symposium on Remote Sensing of the Environment*, pages 1309–1317, 1977.

[92] M. Kearns and M. Li. Learning in the presence of malicious error. *SIAM Journal of Computing*, 22(4), 1993.

[93] M. Kearns and U. Vazirani. *Introduction to computational Learning Theory.* MIT Press, 1994.

[94] D. G. Kendall. Shape manifolds, procrustean metrics, and complex projective shapes. *Bulletins of the London Mathematical Society*, 16:81–121, 1984.

[95] D. G. Kendall. A survey of the statistical theory of shape. *Statistical Science*, 4(2):87–120, 1989.

[96] C. Kervrann, F. Davoine, P. Perez, H. Li, R. Forchheimer, and C. Labit. Generalized likelihood ratio-based face detection and extraction of mouth features. In *Proceedings of the First International Conference on Audio- and Video-based Biometric Person Authentication*, pages 27–34, 1997.

[97] S.-H. Kim, N.-K. Kim, S. C. Ahn, and H.-G. Kim. Object oriented face detection using range and color information. In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 76–81, 1998.

[98] M. Kirby and L. Sirovich. Application of the Karhunen-Loève procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108, 1990.

[99] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. In *Proceedings of the Annual ACM Symposium on the Theory of Computing*, 1995.

[100] R. Kjeldsen and J. Kender. Finding skin in color images. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 312–317, 1996.

[101] T. Kohonen. *Self Organizing Map.* Springer, 1996.

[102] C. Kotropoulos and I. Pitas. Rule-based face detection in frontal views. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 21–24, 1997.

[103] C. Kotropoulos, A. Tefas, and I. Pitas. Frontal face authentication using variants of dynamic link matching based on mathematical morphology. In *Proceedings of IEEE International Conference on Image Processing*, pages 122–126, 1998.

[104] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[105] E. Kushilevitz and D. Roth. On learning visual concepts and DNF formulae. *Machine Learning*, 24(1):65–85, 1996.

[106] Y. H. Kwon and N. da Vitoria Lobo. Face detection using templates. In *International Conference on Pattern Recognition*, pages 764–767, 1994.

[107] K. Lam and H. Yan. Fast algorithm for locating head boundaries. *Journal of Electronic Imaging*, 3(4):351–359, 1994.

[108] A. Lanitis, C. J. Taylor, and T. F. Cootes. An automatic face identification system using flexible appearance models. *Image and Vision Computing*, 13(5):393–401, 1995.

[109] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems 2*, pages 396–404, 1990.

[110] T. Leung, M. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. In *Proceedings of the Fifth IEEE International Conference on Computer Vision*, pages 637–644, 1995.

[111] M. S. Lew. Information theoretic view-based and modular face detection. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 198–203, 1996.

[112] F. Leymarie and M. D. Levine. Tracking deformable objects in the plan using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):617–634, 1993.

[113] J. J.-J. Lien, T. Kanade, J. F. Cohn, and C.-C. Li. Subtly different facial expression recognition and expression intensity estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 853–859, 1998.

[114] S.-H. Lin, S.-Y. Kung, and L.-J. Lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE Transactions on Neural Network*, 8(1):114–132, 1997.

[115] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[116] N. Littlestone. Redundant noisy attributes, attribute errors, and linear threshold learning using winnow. In *Proceedings of the fourth Annual Workshop on Computational Learning Theory*, pages 147–156, 1991.

[117] M. M. Loève. *Probability Theory*. Van Nostrand, Princeton, 1955.

[118] A. C. Loui, C. N. Judice, and S. Liu. An image database for benchmarking of automatic face detection and recognition algorithms. In *Proceedings of IEEE International Conference on Image Processing*, pages 146–150, 1998.

[119] O. L. Mangasarian and D. R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10(5):1032–1037, 1999.

[120] O. L. Mangasarian, R. Setiono, and W. Wolberg. Pattern recognition via linear programming: Theory and application to medical diagnosis. In T. F. Coleman and Y. Li, editors, *Large-scale numerical optimization*, pages 22–30. SIAM Publications, 1990. data available at ftp://128.195.1.46/ pub/ machine- learning- databases/.

[121] K. V. Mardia and I. L. Dryden. Shape distributions for landmark data. *Advanced Applied Probability*, 21:742–755, 1989.

[122] A. Martinez and R. Benavente. The AR face database. Technical Report CVC Technical Report 24, Purdue University, 1998.

[123] K. Matsuno, C.-W. Lee, S. Kimura, and S. Tsuji. Automatic recognition of human facial expressions. In *Proceedings of the Fifth IEEE International Conference on Computer Vision*, pages 352–359, 1995.

[124] S. McKenna, S. Gong, and Y. Raja. Modelling facial colour and identity with Gaussian mixtures. *Pattern Recognition*, 31(12):1883–1892, 1998.

[125] S. McKenna, Y. Raja, and S. Gong. Tracking colour objects using adaptive mixture models. *Image and Vision Computing*, 1998.

[126] G. J. McLachlan. Assessing the performance of an allocation rule. *Comp. & Maths. with Appls.*, 12A:261–272, 1986.

[127] G. J. McLachlan and K. E. Basford. *Mixture Models: Inference and Applications to Clustering.* Marcel Dekker, 1988.

[128] J. Miao, B. Yin, K. Wang, L. Shen, and X. Chen. A hierarchical multiscale and multiangle system for human face detection in a complex background using gravity-center template. *Pattern Recognition*, 32(7):1237–1248, 1999.

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

[130] B. Moghaddam and A. Pentland. Probabilistic visual learning for object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, 1997.

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

[132] S. K. Nayar, S. A. Nene, and H. Murase. Real-time 100 object recognition system. In *Proceedings of IEEE International Conference on Robotics and Automation*, 1996.

[133] A. V. Nefian and M. H. H. III. Face detection and recognition using Hidden Markov Models. In *Proceedings of IEEE International Conference on Image Processing*, volume 1, pages 141–145, 1998.

[134] N. Oliver, A. Pentland, and F. Berard. LAFER: Lips and face real time tracker. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 123–129, 1997.

[135] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 193–199, 1997.

[136] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[137] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proceedings of the Sixth IEEE International Conference on Computer Vision*, pages 555–562, 1998.

[138] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[139] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *Proceedings of the Fourth IEEE International Conference on Computer Vision*, pages 84–91, 1994.

[140] P. J. Phillips, H. Moon, P. J. Rauss, and S. Rizvi. The FERET evaluation methodology for face recognition algorithms. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 137–143, 1997.

[141] P. J. Phillips, H. Moon, S. Rizvi, and P. Rauss. The FERET evaluation. In H. Wechsler, P. J. Phillips, V. Bruce, F. Fogelman-Soulie, and T. S. Huang, editors, *Face Recognition: From Theory to Applications*, volume 163 of *NATO ASI Series F, Computer and Systems Sciences*, pages 244–261. Springer, 1998.

[142] P. J. Phillips, P. J. Rauss, and S. Z. Der. FERET (face recognition technology) recognition algorithm development and test report. Technical Report ARL-TR-995, U.S. Army Research Laboratory, 1996.

[143] S. Pigeon and L. Vandendrope. The M2VTS multimodal face database. In *Proceedings of the First International Conference on Audio- and Video-based Biometric Person Authentication*, 1997.

[144] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods*, chapter 12, pages 185–208. MIT Press, 1998.

[145] T. Poggio and S. Edelman. A network that learns to recognize three-dimensional objects. *Nature*, 343:263–266, 1990.

[146] D. A. Pomerleau. Knowledge-based training of artificial neural networks for autonomous robot driving. In J. Connell and S. Mahadevan, editors, *Robot Learning*, pages 19–43. Kluwer Academic Publishers, 1993.

[147] M. Pontil and A. Verri. Support vector machines for 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):637–646, 1998.

[148] F. P. Preparata and M. I. Shamos. *Computational Geometry*. Springer-Verlag, 1985.

[149] M. Propp and A. Samal. Artificial neural network architectures for human face detection. *Intelligent Engineering Systems Through Artificial Neural Networks*, 2, 1992.

[150] R. J. Qian and T. S. Huang. Object detection using hierarchical MRF and MAP estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 186–192, 1997.

[151] R. J. Qian, M. I. Sezan, and K. E. Matthews. A robust real-time face tracking algorithm. In *Proceedings of IEEE International Conference on Image Processing*, pages 131–135, 1998.

[152] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Kluwer Academic, 1993.

[153] A. Rajagopalan, K. Kumar, J. Karlekar, R. Manivasakan, M. Patil, U. Desai, P. Poonacha, and S. Chaudhuri. Finding faces in photographs. In *Proceedings of the Sixth IEEE International Conference on Computer Vision*, pages 640–645, 1998.

[154] R. A. Render and H. F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.

[155] S. A. Rizvi, P. J. Phillips, and H. Moon. A verification protocol and statistical performance analysis for face recognition algorithms. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 833–838, 1998.

[156] D. Roobaert and M. V. Hulle. View-based 3D object recognition with support vector machines. In *IEEE International Workshop on Neural Networks for Signal Processing*, 1999.

[157] D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of National Conference on Artificial Intelligence*, 1998.

[158] D. Roth, M.-H. Yang, and N. Ahuja. Learning to recognize objects. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[159] H. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. In *Advances in Neural Information Processing Systems 8*, pages 875–881, 1996.

[160] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 203–208, 1996.

[161] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[162] H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 38–44, 1998.

[163] H. A. Rowley. *Neural Network-Based Face Detection*. PhD thesis, Carnegie Mellon University, 1999.

[164] P. Ruján. Playing billiards in version space. *Neural Computation*, 9(1):99–122, 1997.

[165] P. Ruján. Computing the Bayes support vector machine. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Large Margin Classifiers*. MIT Press, 1999.

[166] E. Saber and A. M. Tekalp. Frontal-view face detection and facial feature extraction using color, shape and symmetry based cost functions. *Pattern Recognition Letters*, 17(8):669–680, 1998.

[167] T. Sakai, M. Nagao, and S. Fujibayashi. Line extraction and pattern detection in a photograph. *Pattern Recognition*, 1:233–248, 1969.

[168] A. Samal and P. A. Iyengar. Automatic recognition and analysis of human faces and facial expressions: A survey. *Pattern Recognition*, 25(1):65–77, 1992.

[169] A. Samal and P. A. Iyengar. Human face detection using silhouettes. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(6):845–867, 1995.

[170] F. Samaria and S. Young. HMM based architecture for face identification. *Image and Vision Computing*, 12:537–583, 1994.

[171] F. S. Samaria. *Face Recognition Using Hidden Markov Models.* PhD thesis, University of Cambridge, 1994.

[172] S. Satoh, Y. Nakamura, and T. Kanade. Name-it: Naming and detecting faces in news videos. *IEEE Multimedia*, 6(1):22–35, 1999.

[173] D. Saxe and R. Foulds. Toward robust skin identification in video images. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 379–384, 1996.

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

[175] B. Schölkopf. *Support Vector Learning.* PhD thesis, Informatik der Technischen Universitat Berlin, 1997.

[176] H. Shvaytser. Learnable and nonlearnable visual concepts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):459–466, May 1990.

[177] P. Sinha. *Processing and recognizing 3D forms.* PhD thesis, Massachusetts Institute of Technology, 1995.

[178] S. A. Sirohey. Human face segmentation and identification. Technical Report CS-TR-3176, University of Maryland, 1993.

[179] J. M. Siskind and Q. Morris. A maximum-likelihood approach to visual event classification. In *Proceedings of the Fourth European Conference on Computer Vision*, pages 347–360, 1996.

[180] J. Sobottka and I. Pitas. Segmentation and tracking of faces in color images. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 236–241, 1996.

[181] K. Sobottka and I. Pitas. Face localization and feature extraction based on shape and color information. In *Proceedings of IEEE International Conference on Image Processing*, pages 483–486, 1996.

[182] F. Soulie, E. Viennet, and B. Lamy. Multi-modular neural network architectures: Pattern recognition applications in optical character recognition and human face recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):721–755, 1993.

[183] T. Starner and A. Pentland. Real-time ASL recognition from video using HMM's. Technical Report Technical Report 375, Media Lab, MIT, 1996.

[184] Y. Sumi and Y. Ohta. Detection of face orientation and facial components using distributed appearance modeling. In *Proceedings of the First International Workshop on Automatic Face and Gesture Recognition*, pages 254–259, 1995.

[185] Q. B. Sun, W. M. Huang, and J. K. Wu. Face detection based on color and local symmetry information. In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 130–135, 1998.

[186] K. Sung and T. Poggio. Example-based learning for view-based human face detection. Technical Report AIM-1521, MIT AI Lab, 1994.

[187] K.-K. Sung. *Learning and Example Selection for Object and Pattern Detection.* PhD thesis, MIT AI Lab, 1996.

[188] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.

[189] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[190] D. L. Swets and J. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):891–896, 1996.

[191] M. Tabb and N. Ahuja. 2-D motion estimation by matching a multiscale set of region primitives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997. submitted.

[192] B. Takacs and H. Wechsler. Face location using a dynamic model of retinal feature extraction. In *Proceedings of the First International Workshop on Automatic Face and Gesture Recognition*, pages 243–247, 1995.

[193] A. Tefas, C. Kotropoulos, and I. Pitas. Variants of dynamic link architecture based on mathematical morphology for frontal face authentication. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 814–819, 1998.

[194] J. C. Terrillon, M. David, and S. Akamatsu. Automatic detection of human faces in natural scene images by use of a skin color model and invariant moments. In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 112–117, 1998.

[195] J. C. Terrillon, M. David, and S. Akamatsu. Detection of human faces in complex scene images by use of a skin color model and invariant Fourier-Mellin moments. In *International Conference on Pattern Recognition*, pages 1350–1355, 1998.

[196] D. M. Titterington, A. F. M. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions.* Wiley, New York, 1985.

[197] A. Tsukamoto, C.-W. Lee, and S. Tsuji. Detection and tracking of human face with synthesized templates. In *Proceedings of the First Asian Conference on Computer Vision*, pages 183–186, 1993.

[198] A. Tsukamoto, C.-W. Lee, and S. Tsuji. Detection and pose estimation of human face with synthesized image models. In *International Conference on Pattern Recognition*, pages 754–757, 1994.

[199] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[200] S. Ullman and S. Soloviev. Computation of pattern invariance in brain-like structures. *Neural Networks*, 12:1021–1036, 1999.

[201] R. Vaillant, C. Monrocq, and Y. Le Cun. An original approach for the localisation of objects in images. In *IEE Proceedings of Vision, Image and Signal Processing*, volume 141, pages 245–250, 1994.

[202] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[203] V. Vapnik. *The nature of statistical learning theory.* Springer, 1995.

[204] M. Venkatraman and V. Govindaraju. Zero crossings of a non-orthogonal wavelet transform for object location. In *Proceedings of IEEE International Conference on Image Processing*, volume 3, pages 57–60, 1995.

[205] C. Vogler and D. Metaxas. ASL recognition based on a coupling between HMMs and 3d motion analysis. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 363–369, 1998.

[206] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3):328–339, 1989.

[207] H. Wang and S.-F. Chang. A highly efficient system for automatic face region detection in MPEG video. *IEEE Transaction on Circuits and Systems for Video Technology*, 7(4):615–628, 1997.

[208] S. Wilcox. Representation of the dynamic elements of signs: Issues in the development of the multimedia dictionary of American sign language. *Journal of Contemporary Legal Issues*, 6, 1995.

[209] A. D. Wilson and A. F. Bobick. Recognition and interpretation of parametric gesture. In *Proceedings of the Sixth IEEE International Conference on Computer Vision*, pages 329–336, 1998.

[210] C. Wong, D. Kortenkamp, and M. Speich. A mobile robot that recognizes people. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*, pages 346–353, 1995.

[211] H. Wu, Q. Chen, and M. Yachida. Face detection from color images using a fuzzy pattern matching method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(6):557–563, 1999.

[212] H. Wu, T. Yokoyama, D. Pramadihanto, and M. Yachida. Face and facial feature extraction from color image. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 345–350, 1996.

[213] G. Yang and T. S. Huang. Human face detection in complex background. *Pattern Recognition*, 27(1):53–63, 1994.

[214] J. Yang, R. Stiefelhagen, U. Meier, and A. Waibel. Visual tracking for multimodal human computer interaction. In *Proceedings of ACM Human Factors in Computing Systems (CHI 98)*, pages 140–147, 1998.

[215] J. Yang and A. Waibel. A real-time face tracker. In *Proceedings of the Third Workshop on Applications of Computer Vision*, pages 142–147, 1996.

[216] M.-H. Yang and N. Ahuja. Detecting human faces in color images. In *Proceedings of IEEE International Conference on Image Processing*, volume 1, pages 127–130, 1998.

[217] M.-H. Yang and N. Ahuja. Extracting gestural motion trajectories. In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 10–15, 1998.

[218] M.-H. Yang and N. Ahuja. Extraction and classification of motion patterns for hand gesture recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 892–897, 1998.

[219] M.-H. Yang and N. Ahuja. Gaussian mixture model for human skin color and its application in image and video databases. In *Proceedings of the SPIE: Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 458–466, 1999.

[220] M.-H. Yang and N. Ahuja. Recognizing hand gestures using motion trajectories. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 466–472, 1999.

[221] M.-H. Yang and N. Ahuja. A geometric approach to train support vector machines. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[222] M.-H. Yang, N. Ahuja, and D. Kriegman. Face detection using a mixture of factor analyzers. In *Proceedings of IEEE International Conference on Image Processing*, 1999.

[223] M.-H. Yang, N. Ahuja, and D. Kriegman. Mixtures of linear subspaces for face detection. In *Proceedings of the Fourth International Conference on Automatic Face and Gesture Recognition*, pages 70–76, 2000.

[224] M.-H. Yang, N. Ahuja, and D. Roth. View-based 3D object recognition with SNoW. In *Proceedings of the Fourth Asian Conference on Computer Vision*, volume 2, pages 830–835, 2000.

[225] M.-H. Yang, D. Roth, and N. Ahuja. Learning to recognize 3D objects with snow. In *Proceedings of the Sixth European Conference on Computer Vision*, 2000.

[226] M.-H. Yang, D. Roth, and N. Ahuja. A SNoW-based face detector. In *NIPS-12; The 1999 Conference on Advances in Neural Information Processing Systems*, pages 855–861. MIT Press, 2000.

[227] K. C. Yow and R. Cipolla. A probabilistic framework for perceptual grouping of features for human face detection. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 16–21, 1996.

[228] K. C. Yow and R. Cipolla. Feature-based human face detection. *Image and Vision Computing*, 15(9):713–735, 1997.

[229] K. C. Yow and R. Cipolla. Enhancing human face detection using motion and active contours. In *Proceedings of the Third Asian Conference on Computer Vision*, pages 515–522, 1998.

[230] A. Yuille, P. Hallinan, and D. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8(2):99–111, 1992.

[231] W. Zhao, R. Chellappa, and A. Krishnaswamy. Discriminant analysis of principal components for face recognition. In *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*, pages 336–341, 1998.

[232] W. Zhao, A. Krishnaswamy, R. Chellappa, D. L. Swets, and J. Weng. Discriminant analysis of principal components for face recognition. In H. Wechsler, P. J. Phillips, V. Bruce, F. Fogelman-Soulie, and T. S. Huang, editors, *Face Recognition: From Theory to Applications*, volume 163 of *NATO ASI Series F, Computer and Systems Sciences*, pages 73–85. Springer, 1998.

Vita

Ming-Hsuan Yang received his B.S. in Computer Science and Power Mechanical Engineering from National Tsing-Hua University, Hsin-Chu, Taiwan, a master degree in Computer Science from University of Southern California and a master degree in Operations Research from University of Texas at Austin. Since 1996, he has been studying Computer Vision with Narendra Ahuja and David Kriegman, and machine learning with Dan Roth as a Ph.D. student in Computer Science Department of University of Illinois at Urbana-Champaign. His research interests include Computer Vision, Human Computer Interaction, Computer Graphics, Machine Learning and Bioinformatics. In 1999, he received the Ray Ozzie Fellowship from University of Illinois at Urbana-Champaign.