A Selective Attention Driven Two-Stage Classifier for Fast Face Detection

A SELECTIVE ATTENTION DRIVEN TWO-STAGE CLASSIFIER FOR FAST FACE DETECTION

By

STEVEN R. JONES, B.Eng. (Electrical Engineering)
McMaster University, Hamilton, Ontario, Canada

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree
Master of Applied Science

McMaster University

MASTER OF APPLIED SCIENCE (2004) MCMASTER UNIVERSITY

(Electrical and Computer Engineering) Hamilton, Ontario

TITLE: A Selective Attention Driven Two-Stage Classifier for Fast Face Detection

AUTHOR: Steven R. Jones

B.Eng. (Electrical Engineering)

McMaster University, Hamilton, Ontario, Canada

SUPERVISOR: Dr. David W. Capson

NUMBER OF PAGES: xvi, 97

Abstract

A novel system for face detection in images and video sequences is presented. The system incorporates a two-stage linear discriminant and nonlinear support vector machine classifier driven by a front-end selective attention search scheme. Additionally, by organizing the system into a client/server framework, efficient parallelization is achieved since the classifier can work independent of the selective attention mechanism once sufficient initial data has been shared between the client and server. Experimental results based on the CMU face image set demonstrate that by using such a classifier arrangement with a non-exhaustive searching scheme, a significant reduction in computational complexity is achieved while maintaining comparable accuracy to other leading face detection systems.

Acknowledgements

I express my sincere gratitude to everyone who helped me complete this thesis. I notably thank my supervisor Dr. David Capson for his total support provided over the past two years.

Notation and Acronyms

Symbol and Definition	
x	Scalar
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
$\mathbf{X}^{\mathrm{T}}$	Matrix transpose
$\|\|\cdot\|\|$	Euclidean 2-norm of vector
C	SVM misclassification tolerance parameter
$\mathcal{D}$	Dataset
$K(x, x')$	Kernel function
$y \in \{-1, +1\}$	Target output values (-1 ='*face*', $+1$ ='*non-face*')
C^*	Optimal misclassification tolerance parameter
γ^*	Optimal Gaussian RBF kernel parameter
k^*	Optimal number of retained eigenvalues
Acronyms	
CVA	Cross Validation Accuracy
ERM	Empirical Risk Minimization
FLD	Fisher Linear Discriminant
FN	False Negative
FP	False Positive

KPCA	Kernel Principal Component Analysis
LDA	Linear Discriminant Analysis
LSVM	Linear Support Vector Machine
MSE	Mean Squared Error
PCA	Principal Component Analysis
RBF	Radial Basis Function
ROC	Receiver Operating Characteristics
SV	Support Vector
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

Contents

Abstract iii

Acknowledgements iv

Notation and Acronyms v

1 Introduction 1

1.1 The Face Detection Problem 2

1.1.1 Challenges 2

1.1.2 Approaches 3

1.1.2.1 Feature-based Approaches 4

1.1.2.2 Image-based Approaches 5

1.1.2.3 Support Vector Machine Based Face Detection . . . 8

1.1.2.4 Real-time Face Detection Systems 10

1.1.2.5 Benchmarking Face Detection Systems 10

1.2 Thesis Overview 11

2 Support Vector Machine-Based Classification 13

2.1 An Overview of Machine Learning 13
2.2 Linear Classification 18
2.2.1 Fisher's Linear Discriminant 19
2.3 Support Vector Machines 20
2.3.1 Linear Support Vector Machines 20
2.3.2 Non-linear SVMs 25
2.4 Improving SVM Classification 30
2.4.1 Bootstrapping 30
2.4.2 k-fold Cross-validation 31
2.4.3 Ensembling 32
2.5 Combining Classifiers 34

3 Appearance-based Modeling of the Face Class **36**

3.1 Image Space Representation 36
3.2 Definition of Dimensionality Reduction 38
3.3 Eigenspace Representation of Face Images 39
3.3.1 Principal Component Analysis 39
3.3.1.1 Eigenspace Computations 42
3.3.2 Modeling the Face Class in Eigenspace 42
3.3.3 Limitations of Principal Component Analysis 46
3.4 Improving Face Class Modeling with Image Preprocessing 47
3.4.1 Spatial Filtering 47
3.4.2 Histogram Equalization 47

3.5 Other Feature Generation Methods 49
3.5.1 Linear Discriminant Analysis 49
3.5.2 Sub-sampled Images 50
3.5.3 Kernel PCA and LDA 51
3.6 Chapter Summary 52

4 Selective Attention Methods for Locating Candidate Face Regions 53

4.1 Visual Attentive Cues 53
4.1.1 Motion 55
4.1.1.1 Spatio-temporal Differencing 55
4.1.1.2 Temporal Integration 57
4.1.2 Saliency Map Creation from Motion 58
4.2 A Biologically-inspired Model for Selective Attention 60
4.2.1 Model 61
4.2.2 Incorporation into a Face Detection Framework 64

5 System Evaluation 67

5.1 System Overview 67
5.1.1 Client-Server Arrangement 69
5.1.2 Selective Attention Server 70
5.1.3 Classification 73
5.1.3.1 Image Preprocessing and Feature Extraction 74
5.1.3.2 Training 75

5.2 Multiscale Searching 77

5.2.1 Improving Results with Overlap Detection 78

5.3 Final Results 82

6 Discussion **85**

A Receiver Operating Characteristic Graphs **88**

List of Tables

2.1 Common kernels used for SVM learning 28

5.1 Selective attention results for CMU test set [43] using 10 attention shifts per image. 71

5.2 Selective attention results for CMU test set [43] using 30 attention shifts per image. 71

5.3 Results comparing motion-based and biologically-inspired selective attention. 73

5.4 Two-stage classifier training procedure. 76

5.5 Comparison of performance results for face detection systems. 83

5.6 System comparison of total windows examined based on the CMU test set. 84

List of Figures

1.1 Typical example of an image containing faces of various scales and poses 2

1.2 Taxonomy of face detection approaches. 3

2.1 Statistical learning process model 15

2.2 Possible seperating hyperplanes 19

2.3 Seperating hyperplanes. (a) One possible separating hyperplane. (b) Hyperplane which maximally separates the two classes. 21

2.4 Geometry of maximum margin hyperplane 22

2.5 Linear seperation in high dimensional space via a nonlinear mapping 26

2.6 Two-dimensional classification example. (a) Using a mapping ϕ, separation in feature space can be found using a linear hyperplane. (b) In the original input space, a corresponding nonlinear ellipsoidal decision boundary is created (Figure modified from [45]). 26

2.7 Synthetic example of 75 points in $\mathbb{R}^2$. (top) A gaussian RBF kernel was used with parameters $\sigma = 1, C = 100$ giving 46 SVs. (bottom) A second order polynomial kernel was used with $C = 100$ giving 64 SVs. Note that the support vectors found by the algorithm are not centers of clusters, but rather examples which are critical for the given classification task. The background shading codes the modulus of the argument of the decision function in equation (2.15) 29

3.1 PCA in action. (a) Synthetic two-class data set. (b) The axes show the first and second principal components along the direction of maximum variance. (c) Data is rotated such that the PC's lie along the axes. (d) The second eigenvalue is removed and the original data is projected onto the first principal component axes. 40

3.2 The 30 most significant eigenfaces ordered by decreasing variance from a set of frontal-view face images [1] are shown. For example, the top-left eigenface represents the most significant principle axis which is the DC component or the mean face image of the data set. 42

3.3 Magnitude of first 60 eigenvalues for face image set [1]. 44

3.4 30 dimensional eigenspace representation of a face image set [1]. 500 faces are plotted in each image showing 3 features. (a) Original raw data set illustrating randomness of 3 features (dimensions 17, 121, 240). (b) First 3 dimensions of eigenspace representation. Notice the variance is largest in these dimensions (c) Dimensions 4, 5 and 6 of the eigenspace representation. (d) Lowest three dimensions of the eigenspace representation. 45

3.5 (a) Quasi-elliptical binary mask operator used. (b) Example face image. (c) Masked face image. 48

3.6 Effect of global histogram equalization on face images. (top) Original face images. (middle) Histogram equalized face images. (bottom) Corresponding histogram of resulting images. 49

3.7 Synthetic example illustrating the class discriminating performance of LDA feature extraction. By projecting the points on $\mathbf{W_{FLD}}$ rather than $\mathbf{W_{PCA}}$, the classes are linearly separable. 50

4.1 Simple demonstration of temporal differencing. (a) Frame 1 of the newscaster sequence [2]. (b) Image differencing of frames 1,2 with T=2. (c) Image differencing of frames 1,2 with T=0. 57

4.2 Illustration of approach 2. (a) Frame 1 of newscaster sequence. (b) Frame 1 of irene sequence. (c) Frame 1 of akiyo sequence. (d) newscaster saliency map, $\beta = 95$, $k = 20$, $T = 65$, pts=129. (e) irene saliency map, $\beta = 95$, $k = 20$, $T = 65$, pts=625. (f) akiyo saliency map, $\beta = 95$, $k = 20$, $T = 65$, pts=48. 59

4.3 Simple example of binary saliency map. Targets are read in a left-to-right, top-to-bottom form 60

4.4 Selective attention model architecture. 63

4.5 sample images from CMU test set after being processed by the front-end selective attention system. 65

5.1 System block diagram overview of face detection system. 68

5.2 Client-server arrangement for face detection system. 69

5.3 sample images from CMU test set after being processed by the front-end selective attention system with 30 acquired targets. The targets are overlayed onto the images as black asterisk symbols. 72

5.4 Sample images from the MIT-CBCL training data set 75

5.5 Relationship between number of support vectors, threshold value and input space dimension based on the initial training data. 78

5.6 Multiscale searching. 79

5.7 ROC curves for two-stage classifier and SVM classifier based on MIT-CBCL test set. 79

5.8 henry image from CMU data set processed without applying overlap detection. Boxes with thick black & white borders indicate multiple hits within a very small region, whereas boxes with thin borders indicate isolated detections. 80

5.9 cnn2600 test image from CMU test set. Minor translations are applied to the reference image. Resulting classifier scores are: (a) 0.2475 (b) 0.1831 (c) 0.0441 (d) 0.0988. 81

5.10 ROC curve comparing two-stage and SVM classifiers based on CMU test set, with user-defined threshold $T_a = -T_b = 0.015$, and $C^* = 2000$, $\gamma^* = 0.42$ and $k^* = 32$ found using eq. (2.20). 82

5.11 Sample images from CMU test set processed by face detection system. 83

A.1 example of ROC curve created by varying the threshold point for a given test set. 91

Chapter 1

Introduction

The human face conveys a great deal of perceptual information and knowledge of its whereabouts has found use in many computer vision applications. Face detection is a necessary front end to face recognition systems with the purpose of localizing and extracting face regions from the background. It has an apparent application in face tracking since it can be used to initiate tracking when a face enters into a frame or appears from an occluded position. It has also found application in areas of content-based image retrieval, object-based video coding, video summarization, video conferencing, crowd surveillance, and intelligent human-computer interfaces.

1.1 The Face Detection Problem

The automatic face[1] detection problem is concerned with finding the coordinate locations of all faces in a given digital image as in Figure 1.1. Although the problem definition is straightforward and clear, a robust and definite solution remains elusive. Over the past 20 years, hundreds of approaches to the problem have emerged in the literature. Very recently the field has expanded rapidly, perhaps in large part due to both the development of powerful statistical machine learning tools and media/government proliferation of security concerns elicited by apparent threats of global terrorism. Even though the field has matured quite significantly, face detection is not a solved problem.

Figure 1.1: Typical example of an image containing faces of various scales and poses

1.1.1 Challenges

Fundamentally, the face detection problem is ill-posed, since no unique mapping exists between the set of all images $\mathbb{Z}^2$ and the set of all face images $\mathrm{F} \in \mathbb{Z}^2$. That is to say, a set of features that uniquely distinguish the class of human faces from all other entities in the universe is unknown. Even if one considers a face to comprise two eyes, one nose and one mouth, the solution is no less vague.

[1]The terms "face" and "human face" will be used interchangeably.

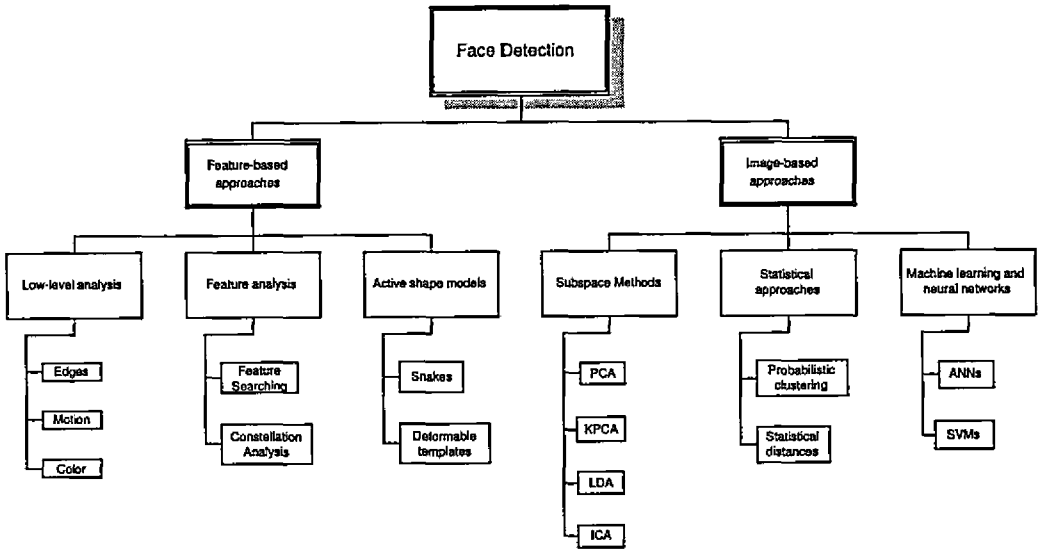


Figure 1.2: Taxonomy of face detection approaches.

The human face is a dynamic object with a high degree of variability in its appearance making face detection a challenging computer vision problem. Common sources of pattern variation include: facial appearance, expression, and presence or absence of common structural features such as glasses, moustache and beard. Additionally, a robust face detector must identify human faces in images at different positions, scales, orientations, pose and lighting condition.

1.1.2 Approaches

Face detection methods can be roughly divided into feature-based and image-based approaches[2]. A complete taxonomy of face detection methods is given in Figure 1.2.

[2]The terms 'image-based' and 'appearance-based' are used interchangeably in the literature.

1.1.2.1 Feature-based Approaches

Many of the early attempts to detect faces used low-level feature-based approaches. Edge detection has been used to analyze line drawings of faces in order to locate facial features (such as eyes, nose and mouth). For example, Craw et al. [11] created a line-follower implemented with curvature constraints to trace a human head outline. Color and intensity information has also been used to extract facial features since eyebrows, lips and pupils generally appear darker than other facial regions. Lam et al. [30] first preprocessed the face images to enhance the local dark patches using contrast stretching and morphological operations. Areas of dark patches are then found using low-level grayscale thresholding.

Motion from video sequences has also been used to create face detection systems. Crowley et al. [13] developed a blink detector as one component in a multi-model face detection system. The fact that both eyes blink together synchronously provides a redundancy cue which permits blinking to be discriminated from other motion in the video sequence. Since the eyes are symmetrically positioned with a fixed separation, the size and orientation of the head can be deduced.

Usually edge, color and motion features extracted from low-level analysis are insufficient and ineffective at detecting complex objects such as faces. Some approaches have improved face detection results by incorporating higher level feature analysis such as face geometry to create stronger verification of ambiguous low-level features. Feature searching techniques aim to determine key facial landmarks; most commonly a pair of eyes. A typical example algorithm starts by heuristically determining the top of the head and then scanning line-by-line to find an eye-plane which appears to contain small, distinct, dark intensity regions. A ratio between the top of the head

and the eye-plane can then be used to infer head size and location.

Another distinct form of feature-based face detection approaches are active-shape models which describe higher-level physical appearances of facial features. Snakes (or active contours) have been used to lock onto head boundaries in order to deduce head shape. Typically the contour is actively constructed iteratively or by minimizing an energy function [48] analogous to a physical system where the internal energy controls the expansion of the snake, and the external energy reflects the ability for the contour to deviate from its current form in order to mold into the shape of head features.

Generally speaking, most feature-based approaches suffer from the unpredictability of facial appearance and imaging conditions; producing mediocre detection results. However, a large amount of literature is available on feature-based methods for frontal-view face detection. For a comprehensive description of methods, see [22], [53].

1.1.2.2 Image-based Approaches

The past decade has witnessed a major paradigm shift in visual recognition. Rather than constructing 3-D shape models or features explicitly, a statistical learning machine can deduce the models automatically through statistical inference from image-based features. For learning to be effective, sufficient training data must be available to allow the machine to account for the large variation in object appearance.

The general procedure for image-based learning approaches to object detection consists of collecting a set of training images of an object. Then (after possibly some preprocessing), the class structure is learned using a distance metric in subspace for

example, or using a statistical classifier to derive a boundary between the object/non-object classes.

Turk and Pentland's landmark paper [50] demonstrated the use of image-based approaches for recognizing faces. Rather than explicitly defining facial features, pixel intensity values were used as features. For an image of 320x240, clearly, a 76800 dimensional feature space is prohibitively large. To reduce the high dimensionality of the feature space, a linear subspace using principal component analysis from a set of training face images is created. The projection of face images into this subspace is termed "eigenspace projection". Then, a measure of "faceness" (called distance from face space (DFFS)) is defined as the Euclidean distance from unknown face pattern $\mathbf{x}$ to the mean vector μ of the eigenface class. In fact, for the recognition problem, M different faces are clustered into M classes in eigenspace. Therefore, an unknown face $\mathbf{x}$ is compared (using DFFS measure) to each of the M classes, declaring a match to the class with the smallest DFFS value. Turk and Pentland's results were important because they demonstrated the effectiveness of deriving features from image intensity values and using linear subspace methods for classifying human faces.

Soon afterwards, Sung and Poggio [47] developed an image-based face detection system demonstrating how the distribution of image patterns from one object (i.e. faces) can be learned from positive and negative example images of that class. Their system consisted of a Gaussian mixture model for face/non-face patterns and a multilayer perceptron classifier. Each face and non-face were preprocessed and sampled as 19x19 images and treated as a 361-dimensional vector. Each vector was grouped into six face and six non-face clusters represented as a Gaussian mixture model. Then, to classify an input pattern, a distance metric (Mahalnobis distance and/or Euclidian distance) is computed between the input pattern and the prototype clusters. Their

learning method places a large focus on discerning between faces and face-like non-faces. They use a modified bootstrapping procedure to generate non-face training examples consisting of only the most *meaningful* non-faces. Their claim is that the system can be made arbitrarily robust because false positive and false negative detection errors can be corrected by iteratively retraining with the appended falsely classified patterns obtained through bootstrapping.

Following from this, Rowley et al. [43] created a frontal-view face detection system based on artificial neural networks (ANN) trained with 20x20 size face images. The network contains three sets of hidden units. One set of units examines quadrants of the images (10x10 regions), one set examines the quadrants of the quadrants (5x5 regions), and one set looks at overlapping 20x5 horizontal strips of the image. With this architecture, each set of hidden units favors detection of local features which might be useful for face detection. For example, the horizontal strips allow detection of mouths or pairs of eyes, while the hidden units with square receptive fields might detect features such as individual eyes, nose or corners of the mouth. The network was trained on a large set of frontal face data (preprocessed to account for varying illumination), and used Sung and Poggio's method for bootstrapping to construct a non-face dataset. Then, faces in an image are found by exhaustively scanning the image at multiple scales and sliding a 20x20 window through the image classifying patterns as face/non-face.

This face detection system has developed into a defacto standard in the field. However, one of the major drawbacks of the neural network approach is that it needs to be extensively tuned (number of hidden layers, nodes, learning rates, etc.) to get excellent results.

1.1.2.3 Support Vector Machine Based Face Detection

Recently, a powerful new machine learning tool called support vector machine (SVM) has evolved due mostly from the work of Vapnik [10]. The concept of support vector machine learning has proven to outperform other machine learning algorithms since SVMs perform structural risk minimization on sparse training data in order to maximize generalization to novel test examples, which, theoretically is superior to other non-parametric learning algorithms based on empirical risk minimization, such as bayes classifier and artificial neural networks [12].

Osuna et al. [38] were the first to use support vector machine (SVM) classification for frontal-view face detection. Their system builds on the work of Poggio [47] and Rowley [43] which exhaustively scans an image for face-like patterns at many possible scales, by dividing the original image into overlapping sub-images and classifying them as a face or non-face. Their SVM classifier was trained with a polynomial kernel using a set of face images (preprocessed similar to Rowley's system) and a set of non-face images obtained through bootstrapping using Sung and Poggio's method. Experiments from the CMU data set [38] demonstrate detection rates of 97.1% and 74.2%, for test sets A and B respectively.

Whereas Rowley's ANN architecture is modeled and tuned specifically for face/non-face classification using three sets of hidden layers, Osuna's SVM approach uses no prior assumptions to compute the decision surface, and can therefore be easily extended to detect other kinds of objects in digital images.

A number of face detection architectures based on SVMs have emerged since Osuna's work. In [27], an ensemble of SVM classifiers was used to improve classification accuracy. A majority vote of the output of several support vector machines is performed

in order to select the most suitable learning machine for detection. Popovici et al. [40] examined the issue of training an SVM in eigenface space. They address the issues of how to select the number of eigenfaces, as well as how to select parameters to achieve good generalization. In [3], a SVM trained on ICA-extracted features was used, with claimed improvement giving 93-97% accuracy using the BANCA database [41].

Most of the proposed techniques up to this point were limited by their inability to handle pose variation in faces. In [37], Heisle proposed a modification to the framework by considering a multi-level SVM system trained on local features (nose, eyes, mouth) rather than a global approach taken by Poggio, Rowley, Osuna and others. Heisle points out that the entire face class under all pose variation does not conform to a compact clustering in the training space and is therefore not well represented by a single SVM. By using a local feature approach, the patterns of local face features vary less under pose changes than the patterns representing the whole face.

Li et al. [32] created a multi-view face detection system by splitting the pose sphere into segments and training a combined eigenface/SVM classifier on each segment. Then, a pose estimator is used to choose the appropriate face detector to make the face/non-face classification. Their results show that an individual support vector machine can handle a variation of $\pm 30^{\circ}$ tilt and $\pm 40^{\circ}$ yaw, with results indicating detection rates of 89-95%.

1.1.2.4 Real-time Face Detection Systems

Practical real-time implementations cannot exhaustively search the image space at all possible scales and locations such as in [43], [38]. Most attempts at reducing search area have integrated visual cues; the most common being skin color [23], [32]. However, this information is not always readily available. In situations where grayscale imaging sensors are being utilized, selective attention is very challenging. One area of investigation in this thesis focuses on front-end non-exhaustive *selective attention* mechanisms that do not rely on color cues.

1.1.2.5 Benchmarking Face Detection Systems

Despite the reportedly high detection rates of most systems, construction of a meaningful benchmark remains difficult due to the absence of a standardized image test set. A few researchers such as Rowley et al. [43] and Popovici et al. [41] have tried to standardize the performance evaluation by publicly distributing their image test sets. The most popular one being the CMU frontal test set [43] which is used in the experiments of chapter 5.

An additional problem with comparing results among face detection systems pertains to how the results are reported. Some researchers have simply reported a single detection percentage along with the number of false positives obtained for a given test set. Recently, a larger number of published systems [33], [42], [43], [52] have presented performance results in the form of a receiver operating characteristic (ROC) curve (see Appendix A) providing a more comprehensive report of how a system performs from a cost/benefit point-of-view under different system parameters.

1.2 Thesis Overview

The work in this thesis builds upon ideas from previous image-based face detection system such as those described above. A new system for frontal-view face detection is developed through the use of a combined classifier system together with a front-end selective attention mechanism aiming to reduce face search space area and significantly speed up the face detection process in grayscale image stills and video sequences.

A number of SVM-based face detection systems have previously been proposed; some of which are described in section 1.1.2.3. The work in this thesis aims to investigate further improvements to SVM learning for face detection and incorporation of non-exhaustive search mechanisms into the process.

A precise list of contributions of this thesis includes:

- Development of a two-stage combined classifier system for face detection based on linear discriminant and support vector machine classifiers.
- Demonstration of how reduced complexity through training and testing can be achieved through such a classifier arrangement.
- A coupling of the face detection system with two different front-end selective attention mechanisms to reduce search space complexity (one based solely on motion, another based on a biologically-inspired model by Itti [26] and very recently investigated in [46] for face detection).
- Performance results for a client/server arrangement of classifier and selective attention components over an adhoc 802.11b wireless network.

- Experimental results based on standardized test data revealing the performance of the overall proposed system.

Chapter 2 offers a theoretical overview of statistical machine learning, linear classification and support vector machines. A few methods for improving classification are presented, and a new combined classifier system is proposed. Chapter 3 discusses appearance-based modelling of the face class by introducing the concept of an image space and the use of manifold projections for learning, as well as the importance of image preprocessing in appearance-based feature extraction. Chapter 4 introduces two techniques for selective attention by integrating motion, intensity and orientation cues into a front-end preprocessor for a face/non-face classification scheme. Chapter 5 presents a description of the proposed integrated face detection framework and provides experimental results of the system benchmarked against common images and video sequences. Chapter 6 summarizes the thesis work and gives a discussion of the obtained results.

Chapter 2

Support Vector Machine-Based Classification

This chapter begins by presenting a brief, foundational overview of statistical learning intending to motivate the theory of support vector machine learning. Next, a detailed account of support vector machine classification is presented along with a description of methods for improving classification accuracy. Finally, a combined two-stage classifier is presented which aims at reducing the overall complexity of SVM-based classification.

2.1 An Overview of Machine Learning

The objective of statistical machine learning from a classification perspective is to find a rule, which, based on external observations, assigns a pattern to one of several

classes.[1] The goal is to estimate an unknown decision[2] function $f : \mathbb{R}^N \rightarrow \{-1, +1\}$ given a finite and usually sparse set of external observations. Let $\mathbf{x} \in \mathcal{X}$ be a random observation feature vector drawn from a finite set of observations $\mathcal{X}$ (for example, a set of face and non-face images). Let $\mathbf{y} \in \mathcal{Y}$ be the interpretation (or target) of $\mathbf{x}$ from a set $\mathcal{Y}$ (for example, labels indicating faces or non-faces) with probability $P(y)$. The joint probability of observing the pair $(\mathbf{x}, y)$ is given by $P(\mathbf{x}, y) = P(\mathbf{x})P(y|\mathbf{x})$ where in practice, $P(\mathbf{x}, y)$ is unknown since the inference function $P(y|\mathbf{x})$ is not known.

However, given the set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^m, \mathbf{x} \in \mathbb{R}^N, y \in \{-1, +1\}$ of observations and corresponding targets, the inference function can be estimated from this data through *inductive learning.* Ultimately, it is the joint probability $P(\mathbf{x}, y)$ that gives complete knowledge of the system being learned.

The general model of inductive learning can be described as having three main components (Figure 2.1) [16]

1. A generator of random vectors (patterns) $\mathbf{x}$, drawn from an i.i.d. source of unknown distribution $P(\mathbf{x})$.
2. A supervisor that returns an output y for every input vector $\mathbf{x}$, according to a fixed, but unknown conditional distribution function $P(y|\mathbf{x})$.
3. A learning machine capable of implementing a set of functions $f(\mathbf{x}, \mathbf{a})$ where $\mathbf{a}$ is a parameter vector.

Under the general statistical machine learning framework, the principle of inductive learning is cast as a problem of function approximation by risk minimization [44].

[1] In the case relevant to the work in this thesis, there are only two different classes defined as $y \in \{-1, +1\}$ and assumed from now on.

[2] The terms 'inference function' and 'decision function' are used interchangeably

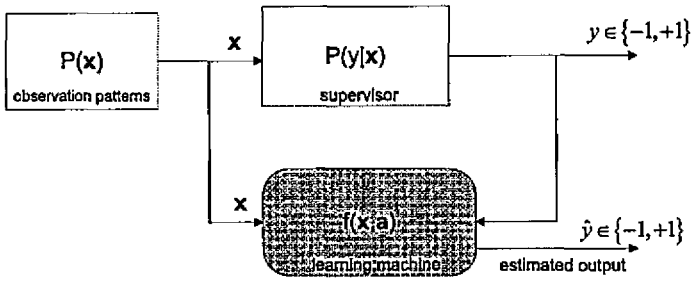


Figure 2.1: Statistical learning process model

That is, a learning machine with some decision function $f(\mathbf{x}, \mathbf{a})$ where $\mathbf{a}$ is a parameter vector, is formed by choosing the function which can *best* approximate the observations with minimal *risk*. This risk function is defined as the expected value of the loss function $L(y, f(\mathbf{x}, \mathbf{a}) = |y - f(\mathbf{x}, \mathbf{a})|$ defined as:

$$\mathcal{R}(a) = \int |y - f(\mathbf{x}, \mathbf{a})| \, P(\mathbf{x}, y) dx dy \tag{2.1}$$

integrated over all possible patterns $(\mathbf{x}, y)$. $\mathcal{R}(a)$ measures the loss or discrepancy between the response y of the supervisor and the response $f(\mathbf{x}, \mathbf{a})$ provided by the learning machine for all given inputs $\mathbf{x}$. Since the risk function is the loss integrated over the unknown joint probability distribution $P(\mathbf{x}, y)$, (2.1) cannot be minimized directly. One common approach is to approximate the risk $\mathcal{R}$, by the *empirical risk* [16]:

$$\mathcal{R}_{emp}(\mathbf{a}) = \frac{1}{m} \sum_{i=1}^{m} |y_i - f(\mathbf{x}_i, \mathbf{a})| \tag{2.2}$$

where m is the number of points in the data set. A learning algorithm which sets its parameters so as to minimize (2.2) performs learning using the principle of *empirical risk minimization* (ERM).

Empirical risk is defined to be the measured mean error rate on the training set (for a fixed, finite number of observations). Many common learning machines use this

criterion for learning. For example, artificial neural networks adjust their weights and biases (using such techniques as the backpropogation algorithm [21] based on minimization of the measured training error rate. Another example includes the classic maximum likelihood method which sets its parameters by minimizing empirical risk in an unsupervised manner using the loss function $L(p(\mathbf{x}, \mathbf{a})) = -log(p(\mathbf{x}, \mathbf{a}))$.

In spite of the widespread application of such learning algorithms based on the ERM principle, it seems apparent that minimizing the training error (or empirical risk) does not necessarily imply a small test error (or risk) averaged over all possible test samples drawn from the underlying distribution $P(\mathbf{x}, y)$. Consequently, a better statistical learning framework is required to achieve low risk and good generalization capability to novel test points. For a given learning task with a specific, finite amount of training data, the best generalization is a tradeoff between the accuracy attained on a *particular* fixed training set and the "*capacity*" of the machine (or the ability of the machine to learn *any* training set without error). In statistical learning theory, this has been termed the *bias-variance problem* . Burges [7] gives a good metaphorical illustration of this dilemma:

> *"A machine with too much capacity is like a botanist with a photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything she has seen before; a machine with too little capacity is like the botanist's lazy brother, who declares that if it's green, it's a tree. Neither can generalize well".*

The mathematical formulation of this concept by Vapnik [51] ultimately showed that a tighter bound (called the VC bound) on the true expected risk holds with probability

of at least $1-\eta$ (for $h < n$, $0 < \eta < 1$)

$$\mathcal{R}(\mathbf{a}) \leq \mathcal{R}_{emp}(\mathbf{a}) + \sqrt{\left(\frac{h\log(\frac{2m}{h})+1) - \log(\frac{\eta}{4})}{m}\right)} \tag{2.3}$$

where h is the non-negative integer called the Vapnik-Chervonenkis (VC) dimension of the classification model, and defines a measure of capacity as described above.

Therefore, given several different learning machines (i.e. family of functions $f(\mathbf{x}, \mathbf{a})$) and for some fixed value η, by taking the machine which minimizes the right-hand side of equation (2.3), the lowest upper bound on the risk is achieved. In other words, equation (2.3) provides a method for estimating how well an algorithm will perform on future data that is solely based on its training set error and a property (VC dimension) of the learning algorithm. In section 2.3 it will be shown that the maximum margin linear support vector machine classifier gives the minimum value for the VC dimension h.

This method for choosing a learning machine is referred to as *structural risk minimization* (SRM), and forms the basis for support vector machine learning described later on in this chapter. A thorough account of SRM and complete proof of equation (2.3) is beyond the scope of this work, but a thorough description is provided in [51]. Nevertheless, the consequence of equation (2.3) is that by choosing a function $f(\mathbf{x}, \mathbf{a})$ with low VC dimension and minimizing its empirical error for a given training set, SRM is a stronger criterion for learning; giving an upper bound (in probability) on the generalization error. Later, the concept of support vector machine learning will be developed demonstrating how the theory of SRM is incorporated into the SVM framework by choosing a low VC dimension.

The concept of inductive learning presented here provides a sound basis and motivation for developing the SVM learning framework described later on in this chapter. However, a discussion of linear discriminant classification is needed to steer the way towards the concept of SVM classification.

2.2 Linear Classification

The family of decision functions for a general linear discriminant classifier has the form $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}+b$, $\mathbf{w} \in \mathbb{R}^N, b \in \mathbb{R}$ and defines a set of decision hyperplanes in $\mathbb{R}^N$. Linear discriminant functions are popular and well developed due to their conceptual and calculable simplicity.

For a linear classifier, given a data set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^m, \mathbf{x} \in \mathbb{R}^N, y \in \{-1, +1\}$, a hyperplane is created such that for a given pattern $\mathbf{x} \in \mathbb{R}^N$, the decision $y \in \{-1, +1\}$ is made such that:

$$\begin{aligned} \mathbf{w}^T\mathbf{x} + b > 0 \quad &\text{for } y = +1 \\ \mathbf{w}^T\mathbf{x} + b < 0 \quad &\text{for } y = -1 \end{aligned} \tag{2.4}$$

Clearly, many criteria exist for determining the parameter $\mathbf{a} = [\mathbf{w}, b]$. More than one hyperplane can potentially separate a given data set as shown in Figure 2.2. The effectiveness of the linear classifier depends on many factors including whether the data is linearly separable or whether the data contains outliers in the training set. For example, the perceptron criterion [44] gives rise to a hyperplane when the data is linearly seperable, and fails to converge to a solution otherwise.

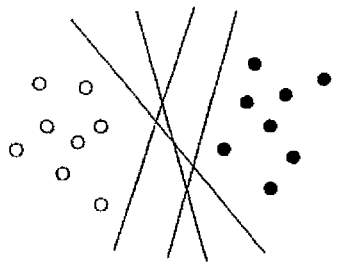


Figure 2.2: Possible seperating hyperplanes

2.2.1 Fisher's Linear Discriminant

Fisher's Linear Disriminant (FLD) [14] is one of the oldest methods of linear discriminant classification. The problem of finding a linear discriminant function is formulated as a problem of minimizing a criterion function. Intuitively, the discriminant hyperplane should be oriented in such a way that the training points of the respective classes are well separated. The FLD defines this "separation" criterion as

$$J(\mathbf{w}) = \frac{|\hat{\mu}_1 - \hat{\mu}_2|^2}{\hat{s}_1^2 + \hat{s}_2^2} \tag{2.5}$$

seeking to maximize the difference of the sample means of the two classes and minimize the sum of the sample variances of the two classes of projected points where $\hat{\mu}_i = \frac{1}{n_i} \sum_{x \in \mathcal{Y}_i} \mathbf{w}^T \mathbf{x}$ is the sample mean for the project points in class $\mathcal{Y}_i$ and $\hat{s}_i = \sum_{y \in \mathcal{Y}_i} (\mathbf{w}^T \mathbf{x} - \hat{\mu}_i)^2$ is the sample variance for class $\mathcal{Y}_i$.

To obtain $J(\mathbf{w})$ as an explicit function of $\mathbf{w}$, the within-class and between-class scatter matrices $S_\mathbf{w}$ and S_B are defined respectively as:

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \tag{2.6}$$

and

$$S_W = \sum_{\mathbf{x} \in C_1} (\mathbf{x} - \mu_1)(\mathbf{x} - \mu_1)^T + \sum_{\mathbf{x} \in C_2} (\mathbf{x} - \mu_2)(\mathbf{x} - \mu_2)^T \tag{2.7}$$

where

$$\mu_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{X}_i} \mathbf{x} \tag{2.8}$$

The fisher criterion is then:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \tag{2.9}$$

and the classic solution to (2.9) is obtained using matrix inversion $\mathbf{w} = S_W^{-1}(\mu_1 - \mu_2)$ [14].

The FLD effectively finds a linear projection to maximally separate the classes. In addition to being a linear disriminant classifier, this notion can be extended to feature extraction by creating a (possibly) lower dimensional subspace with which to project data. Feature extraction is described later in Chapter 3.

2.3 Support Vector Machines

2.3.1 Linear Support Vector Machines

In this section, an alternative criterion for designing a linear classifier is presented based on the theory of structural risk minimization. As discussed briefly in section 2.1, theoretically, in order to achieve lower risk and better generalization of a classifier, not only does the empirical risk (training error) need to be minimized, the capacity needs to be appropriately chosen. The VC dimension described earlier is a popular way to measure and quantify capacity.

In [51], Vapnik showed that for a linear classifier, the VC dimension h is bounded according to $h \leq \|w\|^2 R^2 + 1$ where $\frac{1}{\|w\|}$ is the margin of separation between the two classes and R is the radius of the smallest ball around the training data which is fixed for a given data set. That is, the capacity decreases with increasing margin. It also makes intuitive sense that the linear classifier which maximally separates the two classes of data will also generalize well to novel test points.

This notion motivates the reasoning behind the criteria for calculating the discriminating hyperplane. The principle idea behind the support vector classifier is to find the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ that maximizes the distance between the closest positive and negative training examples as in Figure 2.3(b). That is, given a data set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^m, \mathbf{x} \in \mathbb{R}^N, y \in \{-1, +1\}$, a maximum separating hyperplane is found between the two classes. This distance is called the margin, which from basic geometry and from Figure 2.4, can easily be derived as $d(\mathbf{w}) = \frac{2}{\|\mathbf{w}\|}$. The optimal hyperplane is obtained by maximizing $\frac{2}{\|\mathbf{w}\|}$, or equivalently, minimizing $\|\mathbf{w}\|$.

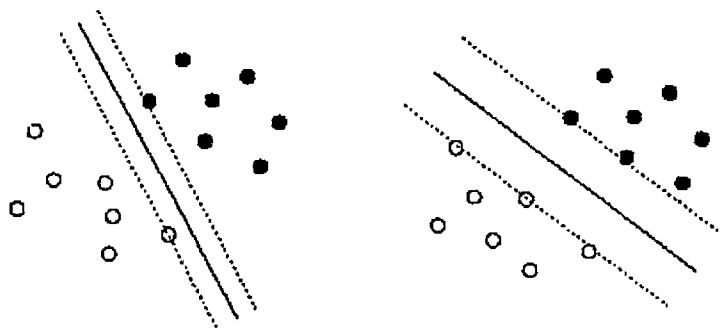


Figure 2.3: Seperating hyperplanes. (a) One possible separating hyperplane. (b) Hyperplane which maximally separates the two classes.

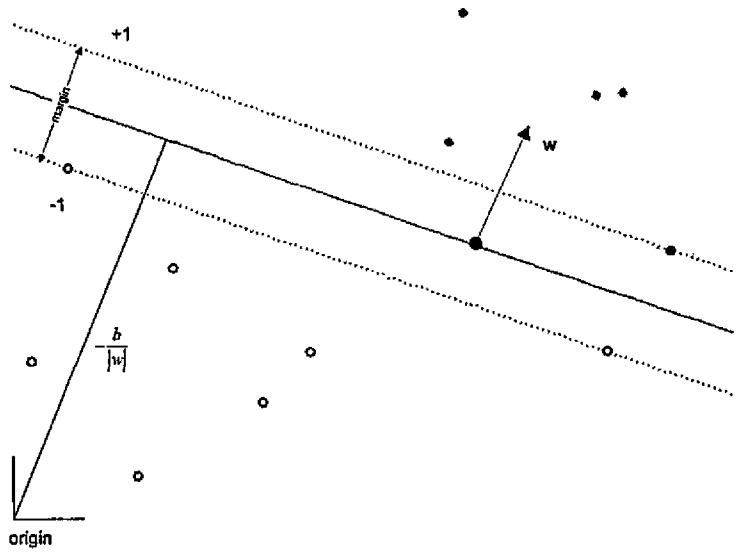


Figure 2.4: Geometry of maximum margin hyperplane

From Figure 2.4, the linear discriminant function must operate such that

$$\begin{aligned} \mathbf{w}^T\mathbf{x} + b > 1, \forall \mathbf{x} \in Class1 \\ \mathbf{w}^T\mathbf{x} + b < -1, \forall \mathbf{x} \in Class2 \end{aligned} \tag{2.10}$$

equivalently, combining the above and writing as an optimization problem:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, i = 1..m \end{aligned} \tag{2.11}$$

In the case where the two classes overlap in the feature space, (2.11) is adjusted by relaxing the constraints allowing some classification errors. Slack variables ξ_i are introduced into the formulation giving $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, i = 1..m$. The problem

now becomes:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, i = 1..m \\ & \xi_i \geq 0 \end{aligned} \tag{2.12}$$

where the added term in the criteria function of (2.12) aims to keep the number of points with ξ_i as small as possible. The variable C is added as a penalty parameter and controls the relative influence of the two competing terms in the cost function. This type of SVM is called a *soft-margin* SVM. Ultimately, C must be chosen to reflect knowledge of the noise on the data which is application dependent. Note that when $C \rightarrow \infty$, (2.12) reduces to the original separable case since the slack variables ξ_i do not enter into the optimization problem explicitly.

Typically, this problem is solved using a different optimization formulation. Using classical Lagrange methods, (2.12) can be converted from the primal problem to its dual problem [12]. This problem is solved by maximization according to a set of Lagrange multipliers α_i:

$$\begin{aligned} \max \quad & W(\alpha) = \sum_{i=1}^{m}\alpha_i - \tfrac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i \cdot \mathbf{x}_j) \\ \text{subject to} \quad & \sum_{i=1}^{m}\alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, i = 1, 2, ..., m \end{aligned} \tag{2.13}$$

In the above formulation, the set of Lagrange multipliers α_i are non-zero for the corresponding "significant" training points. The significant training points are entitled the *support vectors* since they are the points of the two classes which define the maximum margin. Only the support vectors factor into the decision hyperplane; all other points are irrelevant. This idea nicely supports the initial intuition behind

better generalization with structural risk minimization: since the hyperplane is completely determined by the patterns closest to it, the solution does not depend on other unknown examples.

Two apparent advantages of the formulation in (2.13) are evident:

1. The training vectors $\mathbf{x}$ enter into the problem via equality constraints and not inequality constraints, which can be easier to solve [12].
2. The training vectors $\mathbf{x}$ enter into the optimization in pairs (as inner products) meaning that the cost function does not depend explicitly on the input space dimension. As explained later on, this property permits efficient generalization to the case of nonlinearly separable classes.

The optimization problem in (2.13) is a convex quadratic programming problem of α with linear constraints. Very efficient codes have been developed for solving the class of quadratic optimization problems. However, in situations where the training size is very large (ex: thousands of training patterns in very high dimension), the optimization becomes prohibitive. Recently, the Sequential Minimization Optimization (SMO) algorithm by Platt [39] was developed for solving large SVM problems. SVM learning with large training sets is obviated in this thesis using a more efficient two-stage classifier described later on.

From the Lagrange formulation in (2.13), the hyperplane parameter $\mathbf{w}$ is a linear combination of the support vectors

$$\mathbf{w} = \sum_{i=1}^{m} y_i \alpha_i \mathbf{x}_i \tag{2.14}$$

The decision function can then be expressed as:

$$\begin{aligned} f(\mathbf{x}) = & \ \mathbf{w}^T\mathbf{x} + b \\ = & \ \sum_{i=1}^{m} y_i\alpha_i(\mathbf{x}\cdot\mathbf{x}_i) + b \end{aligned} \tag{2.15}$$

where the bias b is computed using the condition $\alpha_i \cdot [y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1] = 0, i = 1..m$.

2.3.2 Non-linear SVMs

In practice, very few problems lend themselves to linear separation in the input pattern space. The linear support vector machine defines an optimal design methodology of a *linear* classifier. Using this construction, the pattern $\mathbf{x}$ under consideration is compared with the SVs $\mathbf{x}_i$ using the inner product as a distance similarity measure, i.e. $\mathbf{x}\cdot\mathbf{x}_i$. However, there is an elegant property of the SVM framework that allows for development of a more general approach by introducing a nonlinear mapping ϕ via a *kernel function* $K(\mathbf{x},\mathbf{x}')$ and performing an inner product measure in the nonlinear space $\mathcal{F}$, i.e. $\phi(\mathbf{x})\cdot\phi(\mathbf{x}_i)$. The purpose of mapping the input features into a possibly higher dimensional space is to satisfactorily separate the classes by a linear hyperplane such as in Figure 2.5.

The following toy example in Figure 2.6 illustrates this idea. Notice how in $\mathbb{R}^2$ a complicated decision function is necessary to separate the two classes. But, by first mapping the points into a feature space $\phi(\mathbf{x})$ as:

$$\begin{aligned} \phi(\mathbf{x}) : \mathbb{R}^2 & \ \mapsto \mathbb{R}^3 \\ (x_1, x_2) & \ \mapsto (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \end{aligned} \tag{2.16}$$

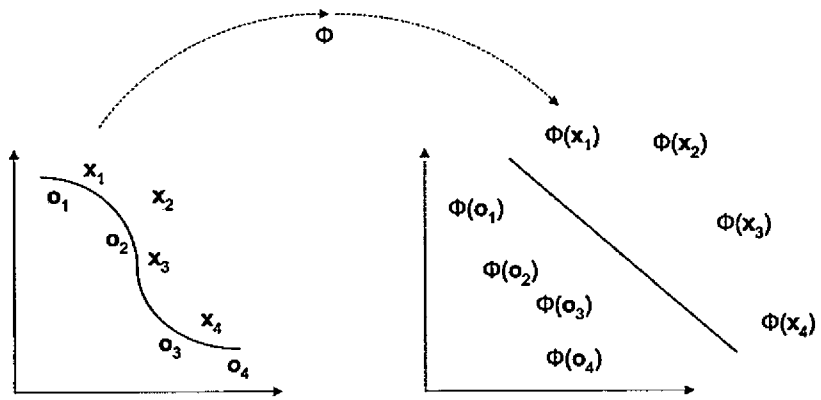


Figure 2.5: Linear seperation in high dimensional space via a nonlinear mapping

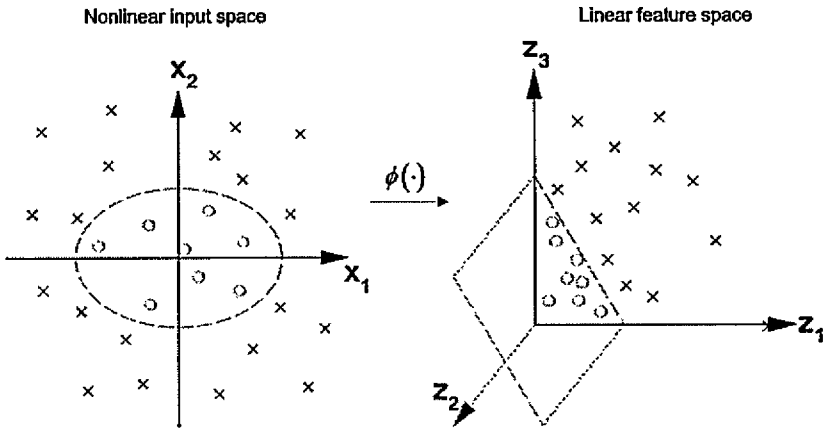


Figure 2.6: Two-dimensional classification example. (a) Using a mapping ϕ, separation in feature space can be found using a linear hyperplane. (b) In the original input space, a corresponding nonlinear ellipsoidal decision boundary is created (Figure modified from [45]).

only a linear hyperplane is required to separate the classes.

In the case of mapping points into a very large (possibly infinite) dimensional space, the SVM model would require the calculation $\phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i)$ which is computationally intensive. Fortunately, there is a highly effective "trick" for computing scalar products in features spaces using kernel functions which satisfy Mercer's Condition [12]. Mercer's theorem provides a condition for when $K(x, x')$ is a kernel. That is, given a data set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^m, \mathbf{x} \in \mathbb{R}^N, y \in \{-1, +1\}$, $K(x, x')$ is a kernel if and only if the matrix

$$\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^m \tag{2.17}$$

is positive semi-definite.

Going back to the previous toy example, the dot product in feature space can be reformulated in terms of a kernel function K:

$$\begin{aligned} (\phi(\mathbf{x}) \cdot \phi(\mathbf{y})) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(y_1^2, \sqrt{2}y_1y_2, y_2^2) \\ &= ((x_1, x_2)(y_1, y_2)^T)^2 \\ &= (\mathbf{x} \cdot \mathbf{y})^2 \\ &= K(\mathbf{x}, \mathbf{y}) \end{aligned} \tag{2.18}$$

requiring a dot product calculation in $\mathbb{R}^2$ rather than in $\mathbb{R}^3$. Therefore, modifiying the SVM concept from equation (2.13) amounts to replacing the inner product $\mathbf{x}_i \cdot \mathbf{x}_j$ with an appropriate kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ satisfying Mercer's condition.

Remarkably, using kernel functions, the scalar product can be implicitly computed in the feature space $\mathcal{F}$ without explicitly using the mapping ϕ, making machine learning in high dimensional feature spaces computationally possible.

The final nonlinear SVM optimization problem is now:

$$\begin{aligned} \max \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & \sum_{i=1}^{m} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, i = 1, 2, ..., m \end{aligned} \tag{2.19}$$

requiring user defined parameters for C and appropriate kernel parameter(s).

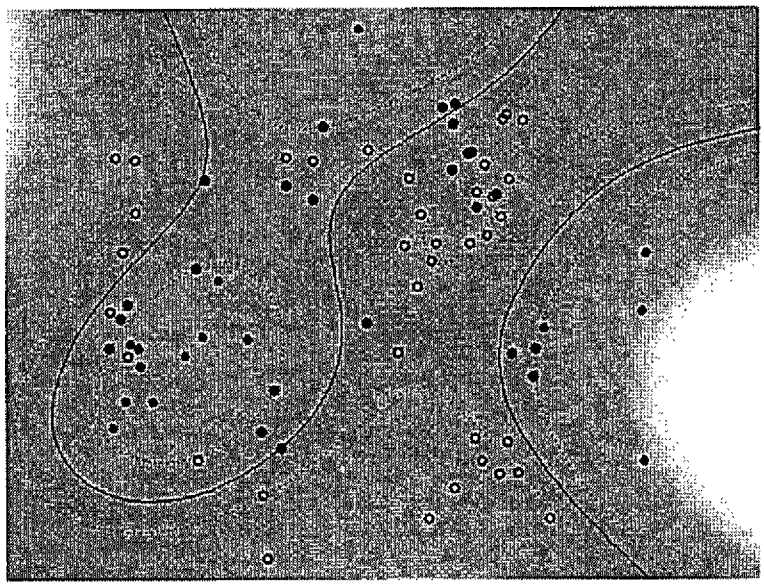
The *proper* choice of kernel function for a given learning function is still an open problem in machine learning. However, a small number of very common kernels have been proposed and used repeatedly in the literature. The most common kernels are shown in Table 2.1.

Linear	$K(x, x') = x \cdot x'$
Polynomial	$K(x, x') = (x \cdot x' + 1)^d$
Gaussian RBF	$K(x, x') = \exp(-\gamma \|x - x'\|^2)$
Sigmoid	$K(x, x') = \tanh(\rho(x \cdot x') + \vartheta)$

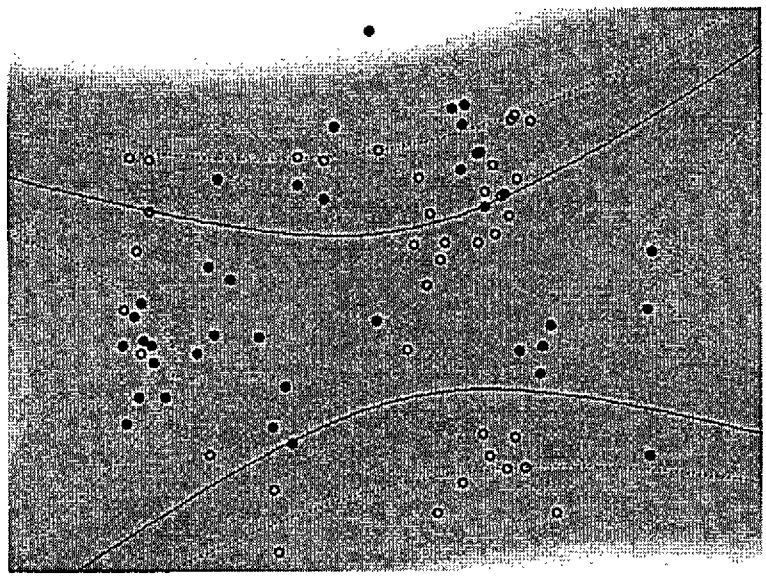
Table 2.1: Common kernels used for SVM learning

To conclude the theoretical development of the SVM, Figure 2.7 illustrates a difficult synthetic classification example created using 75 points in $\mathbb{R}^2$ and solved with nonlinear SVMs. In Figure 2.7(a), a Gaussian RBF kernel was used and in Figure 2.7(b), a second order polynomial was used.

As a final note, the binary support vector machine is readily extendable to the M-class problem. However, this idea is not useful for the applications described in this thesis. Cristianini and Shawe-Taylor [12] provide techniques for extending the SVM to multi-class problems. In this thesis, the two-class problem is assumed where otherwise unnoted.



(a)



(b)

Figure 2.7: Synthetic example of 75 points in $\mathbb{R}^2$. (top) A gaussian RBF kernel was used with parameters $\sigma = 1, C = 100$ giving 46 SVs. (bottom) A second order polynomial kernel was used with $C = 100$ giving 64 SVs. Note that the support vectors found by the algorithm are not centers of clusters, but rather examples which are critical for the given classification task. The background shading codes the modulus of the argument of the decision function in equation (2.15)

2.4 Improving SVM Classification

Three methods are described below for improving the performance of SVM classification. These concepts draw directly from generalized machine learning improvements. In this thesis, a combination of bootstrapping and k-fold cross-validation are used to improve classification and select appropriate model parameters.

2.4.1 Bootstrapping

In general, more training data leads to a better classifier. However, training a classifier with a very large data set can be difficult. Typically only a relatively small number of data points are important for determining the decision function of the classifier. A common technique that takes advantage of this fact is called "bootstrapping" [34]. In machine learning, the term "bootstrapping" refers to the process of iteratively training a classifier on a subset of the total examples and then appending new examples to the set based on errors made by the classifier. Bootstrapping is based on the fact that the false positives and false negatives are the particular training cases that give the most information about errors in the configuration of the classifier.

Bootstrapping for two-class SVM learning can be essential if one or more of the classes encompass a large set of possible patterns. In an object detection framework, usually one class represents the object of interest while the second class represents *everything else*. In this case, it is not feasible to provide a large enough sample of training points in the "non-object" class which is obviously much broader and richer. Although non-object patterns are abundant, it is difficult to characterize and define ones that are *useful* from a learning point of view. Consider the case where the training data is

of dimension MN (using MxN size 8-bit grayscale images). 256^{MN} points enclose the entire space of patterns. For only a small 19x19 image size, this amounts to over 10^{693} possible patterns!

To overcome this dilemma, rather than try to exhaust patterns in the non-object class, bootstrapping can be useful in collecting important patterns which the classifier cannot handle (i.e. misclassifies). A large set of random "non-object" class patterns are used to test the SVM. The set of patterns reported from the classifier as being from the "object" class are collected. These patterns comprise what may be considered "*near-object*" patterns (i.e. patterns close to the object/non-object boundary). Recall that in SVM learning, only the support vectors (or points on the boundary of the two-classes) are significant. The SVM is then retrained using the original training data along with this appended set of bootstrapped patterns. This process can be iterated any number of times until a satisfactorily low number of misclassifications is attained. Osuna et al. [38] used this method to improve classification of faces/non-faces and suggest the use of natural images such as rocks and trees for bootstrapping since they provide many different textured patterns.

More details of this bootstrapping method as it pertains to the experiments performed are provided later in Chapter 5.

2.4.2 k-fold Cross-validation

Cross-validation is mainly used as a statistical tool for deciding between alternative parameterizations given a data set. Therefore it is also a way of checking how well a machine will generalize to new data. Cross-validation is popular in measuring generalization capabilities of neural networks as well as for parameter selection such

as choosing the number of hidden units to select. SVM learning can make use of cross-validation to select optimal model parameters C^* and kernel parameter γ^* as in equation (2.20).

A number of variants of cross-validation exist (k-fold, leave one out, leave v out). In k-fold cross-validation, the training data set $\mathcal{D}$ of m points is randomly partitioned into k subsets of equal size $\frac{k}{m}$. Iteratively $\binom{k}{1}$ times, one subset is tested using the classifier trained on the remaining $k-1$ subsets. Therefore, each instance of the entire training set is predicted once. The classification results are aggregated and a final cross-validation accuracy (CVA) value is computed as the percentage of data correctly classified. CVA provides a quick and reasonable metric for choosing SVM parameters. In this thesis, to determine the "best" parameter set (C, γ), a coarse grid-search over C and γ is used to find

$$(C^*, \gamma^*) = \max_{C,\gamma} CVA \tag{2.20}$$

After identifying a coarse region on the (C, γ) grid, a finer grid search in this region can be conducted. After the best (C^*, γ^*) is found, the whole training set is trained again using these parameters. A more descriptive account of the techniques used in this thesis is given in Chapter 5.

2.4.3 Ensembling

An ensemble is a type of multiple classifier system based on the combination of outputs of several different classifiers intending to obtain higher overall performance of classification. Typically, several classifiers are trained with either different data sets or different learning algorithms. Then, a majority consensus scheme is used to

make a final decision.

Hansen and Salamon [20] showed why an ensemble of classifiers performs better than an individual classifier. Assume that there are an ensemble of N classifiers: $\{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_N(\mathbf{x})\}$ for some test data $\mathbf{x}$. If all the classifiers are different with uncorrelated errors, then when $f_i(\mathbf{x})$ misclassifies $\mathbf{x}$, possibly some or all of the other classifiers may correctly classify $\mathbf{x}$ and will influence the final result by the use of majority voting. Mathematically, if the error for an individual classifier is $p < 0.5$ and the errors are independent for all classifiers, then the probability p_E that the majority voting is incorrect is:

$$p_E = \sum_{k=\lceil N/2 \rceil}^{N} p^k (1-p)^{(N-k)} < \sum_{k=\lceil N/2 \rceil}^{N} \left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{(N-k)} = \sum_{k=\lceil N/2 \rceil}^{N} \left(\frac{1}{2}\right)^N \tag{2.21}$$

and as $N \to \infty$, $p_E \to 0$.

Although Je, Kim and Bang [27] demonstrate how SVM ensembling can be used to improve classification rates for a face detection problem, a high amount of computational overhead may be required to achieve the improved accuracy. The accuracy-complexity tradeoff needs to be assessed depending on the requirements for a particular application. A number of ensembling approaches exist such as: Bagging [4] and Boosting [52]. In the work presented in this thesis, only bootstrapping and cross-validation techniques for improving classification are used.

2.5 Combining Classifiers

So far, a description of SVM learning and methods for improving accuracy have been described. Despite the powerful generalization capabilities of SVMs as described in section 2.1, their main drawback lies in the applicability to real-time classification due to the computational cost incurred relative to other more simplistic classifiers. First, calculation of the kernel function can be costly but seems unavoidable. Second, from equation (2.15), it is apparent that the complexity of the SVM classifier is a function of the number of support vectors.

Burges' [6] method for reducing the number of support vectors by approximating the boundary aims to circumvent the complexity issue. However, it is limited to a specific class of polynomial kernels.

A fundamentally different approach to complexity reduction is developed and investigated in this thesis. Rather than attempting to reduce the number of support vectors by approximating the boundary, SVM learning is simplified by introducing an initial linear classifier (Fisher's linear discriminant classifier) which behaves as a coarse prefilter aiming to conceal a large amount of the input patterns from the nonlinear SVM classifier.

A two-stage combined FLD/SVM classifier model is presented in this thesis which improves overall performance over a single SVM-based detection system in terms of combined speed and accuracy. Under most circumstances, the FLD classifier is computationally more efficient but less accurate than the nonlinear SVM classifier. The advantages of this speed/accuracy tradeoff are combined in the following manner. Given an unknown input $\mathbf{x}$, the FLD classifier produces a value for the decision

function $f(\mathbf{x}) \lessgtr 0$. A more strict set of thresholds T_a and T_b are defined so that if classification of the unknown pattern $\mathbf{x}$ falls into the region $f(\mathbf{x}) \in [-T_a, T_b]$, the pattern is labeled ambiguous and is then passed into the second stage SVM classifier. The values of T_a, T_b are application dependent and reflect the speed/accuracy tradeoff of the system. The classifier rule is therefore:

$$f(\mathbf{x}) = \begin{cases} sgn\left(\mathbf{w}^T\mathbf{x} + b\right) & \text{if } \mathbf{w}^T\mathbf{x} + b \notin [-T_a, T_b] \\ sgn\left(\sum_{i \in SV} y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b\right) & \text{otherwise} \end{cases} \tag{2.22}$$

This process consists of an initial quick detection phase by the FLD classifier followed by a more precise SVM phase. Using the FLD, the linear classifier always has a fixed number of computations $O(d)$ (where d is the dimension of the training space $\mathbb{R}^d$) making it fast and ideal for classification of unambiguous patterns.

With this arrangement, the initial FLD classifier performs as a linear pre-filter for the SVM stage. Therefore, training the nonlinear SVM classifier is simplified since it is only trained on the examples in the ambiguous region as well as some additional bootstrapped images, which simplifies the learning stage and produces a smaller set of support vectors. This in turn speeds up training, since the complexity of non-linear SVM classification is a direct function of the number of support vectors.

One might argue that the SVM may not inherit any decision boundary simplicity from this method since the SVM is still learning the patterns which prove to be too complex for the initial linear classifier. And, since only the points that lie close to the boundary are important, only the points away from this boundary are filtered by the FLD classifier anyway. In Chapter 5, results of this combined classifier are more thoroughly compared with a single SVM-based classifier system through ROC analysis.

Chapter 3

Appearance-based Modeling of the Face Class

An overview of the concept of an image space is given. The classic method of principal component analysis for dimensionality reduction and feature extraction is explained as it pertains to the experiments in Chapter 5. Finally, limitations of the method are described and other feature extraction techniques are briefly discussed.

3.1 Image Space Representation

Appearance-based approaches to vision are built on the concept of an *image space*. A two-dimensional digital image $I(x, y)$ can be viewed as a point (vector) in a very high dimensional space where each spatial coordinate corresponds to a pixel of the image. For example, an image with 19 rows and 19 columns can be described as a point

in a 361-dimensional image space. Using such a representation, the neighborhood relationship amongst pixels in a two-dimensional image is removed. That is, rearranging the pixels in the image (and therefore changing neighborhood relationships) will have no end effect on its image space representation so long as all other images are identically rearranged[1].

Using an image space representation, the image becomes a high dimensional feature allowing traditional feature-based methods in statistical pattern recognition to be used such as clustering and distance metrics. Of course, the high dimensionality of the image space can make many feature based operations impractical, so a careful consideration of data reduction is required.

If an image of an object is a point in image space, then a collection of M images define M points in image space which constitute samples of the probability distribution. Clearly, the size of an image space is enormous. As described in section 2.4.1, for a small 19x19 image, the image space is composed of more than 10^{693} possible coordinates.

The image space representation therefore gives rise to a number of key questions:

- In the case of face images, what can be said about the arrangement of points in image space that correspond to all possible human faces?
- What transformation maps points $\mathbf{x}$ into new feature points $\mathbf{z}$ such that a more "compact" representation of the face space is attained in image space?
- Can the whole face space be efficiently characterized by only a small subset of

[1]As a matter of pure convention and convenience, the image vector is typically formed using a raster scan reading of an image.

sample face training images?

- How can facial lighting variation and small spatial transformations be accounted for in image space?

Turk [49] makes the supposition that images of a particular object (such as human faces) under various transformations (ex: lighting, spatial) occupy a small but distinct region of the image space. However, answers to these questions remain obscure despite a great deal of research in appearance-based computer vision, initiated by Turk and Pentland's landmark paper on eigenfaces for recognition [50].

3.2 Definition of Dimensionality Reduction

Consider a sample of data $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^m$ of N-dimensional vectors lying in a data space $\mathbb{R}^N$. The fundamental assumption that justifies dimensionality reduction is that the sample of points $\mathcal{D}$ at least approximately lie on a manifold of smaller dimension than the data space $\mathbb{R}^N$.

Suppose that in representing a point $\mathbf{x}_i$ using a lower dimensional representation with $k < N$ dimensions, $\mathbf{x}_i$ is simply truncated to dimension k; then the mean-squared error (MSE) will equal the sum of the variances of the elements eliminated from $\mathbf{x}_i$ [21]. Therefore, the goal of dimensionality reduction is to find a representation of a manifold (a coordinate system) that will allow the projection of data vectors on it to obtain a low dimensional compact representation of the data such that the reconstruction error is small. In a MSE sense, principal component analysis gives the optimal projection [21].

3.3 Eigenspace Representation of Face Images

Perhaps the most extensively used approach to addressing some of the issues surrounding image (face) space representation such as the dimensionality problem, is the eigenspace framework [50], [40].

Principal component analysis[2] (PCA) is used to create a linear projection from a collection of sample points $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^m$ such that the projected data lies on the principal axes which are orthonormal and lie in the directions of maximum covariance of the data. The purpose of PCA is to reduce the dimensionality of the data set while retaining as much of the variance as possible. Therefore PCA makes the assumption that the second order statistics define the *useful* features comprising the image space, and that directions corresponding to low variance can be removed; thus limiting the projected space to a lower dimensional manifold.

In Figure 3.1, a simple example is developed to illustrate the previous discussion where PCA finds the basis vectors u_1 and u_2 of maximum variance for a synthetic 2D data set, and finally reduces the dimension to 1D.

3.3.1 Principal Component Analysis

Consider a data set $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^m$ of N-dimensional vectors. In the context of face detection, the data set consists of m face images vectorized in raster-scan format. The mean, μ and the covariance matrix Σ of the data set $\mathcal{D}$ are given by:

[2]also referred to as the Karhuhen-Loeve transform or the Hotelling transform

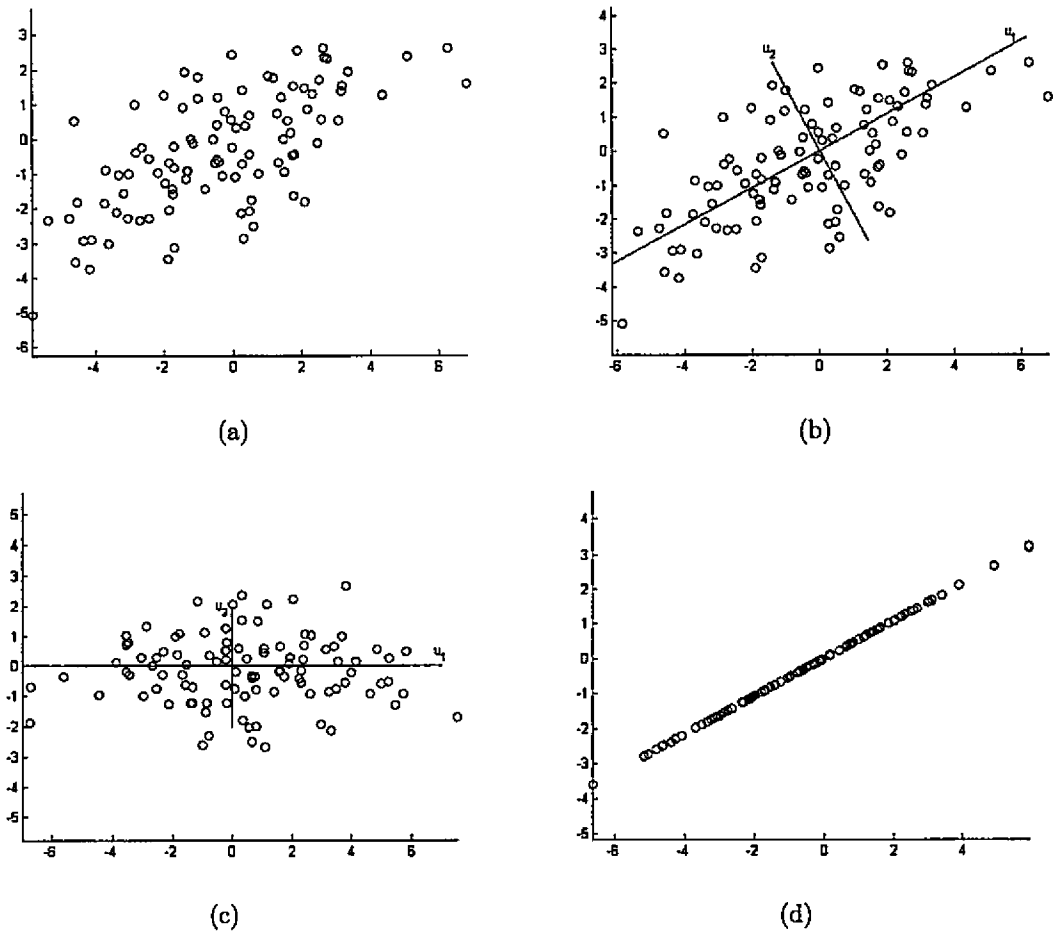


Figure 3.1: PCA in action. (a) Synthetic two-class data set. (b) The axes show the first and second principal components along the direction of maximum variance. (c) Data is rotated such that the PC's lie along the axes. (d) The second eigenvalue is removed and the original data is projected onto the first principal component axes.

$$\begin{aligned} \mu &= \frac{1}{m}\sum_{i=1}^{m} x_i \\ \Sigma &= \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu)(x_i - \mu)^T \end{aligned} \tag{3.1}$$

where the matrix Σ characterizes the scatter of the data set. The eigenvectors $\mathbf{u}_i$ and eigenvalues λ_i of the covariance matrix Σ are found from the standard eigenvalue problem:

$$\Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i \tag{3.2}$$

Then, the eigenvectors $\mathbf{u}_i$ can be sorted according to the size of their corresponding eigenvalues. A large eigenvalue implies that more of the variance in the data is captured by the eigenvalue. Therefore, only the best k eigenvectors (those with the largest k eigenvalues) are retained. The final linear transform is then

$$T(\mathbf{x}) = \mathbf{U}^T(\mathbf{x} - \mu) \tag{3.3}$$

Where $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \ldots, \mathbf{u}_k]$ is the eigenvector matrix and μ is the mean vector of the data set as in equation (3.1). A good thorough account of the derivation and reasoning behind equation (3.2) is given in [21] and [28].

When using face images to represent the vectors $\mathbf{x}$, the eigenvectors $\mathbf{u}_i$ are commonly called *eigenfaces* and can be visually represented since the eigenvectors belong to the same vector space as the face images, and can therefore be viewed as $m \times n$ face images. These ghostly-looking eigenfaces form the basis vectors or the principal axes in the eigenspace. Figure 3.2 shows the first 30 eigenfaces (ordered by decreasing variance) for a face image training set [1] used in the experiments of this thesis.

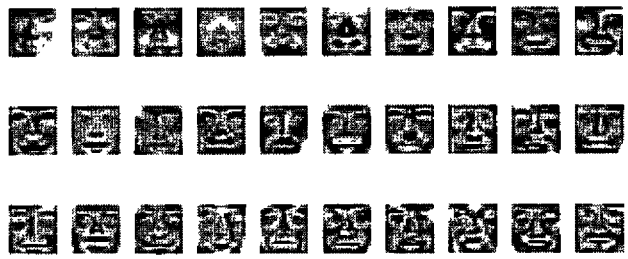


Figure 3.2: The 30 most significant eigenfaces ordered by decreasing variance from a set of frontal-view face images [1] are shown. For example, the top-left eigenface represents the most significant principle axis which is the DC component or the mean face image of the data set.

3.3.1.1 Eigenspace Computations

The critical calculation for building the eigenspace (i.e. the linear mapping matrix **U**) is solving the eigenvalue problem (3.2) for the covariance matrix Σ. Normally Σ will be a huge matrix if the image dimension is much higher than the total number of sample images. In this case, calculating the eigenvectors and eigenvalues is impractical computationally. Fortunately, a relationship exists between Σ and the implicit form of the covariance matrix $\Sigma' = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \mu)^T(\mathbf{x}_i - \mu)$ and is described by Turk [49]. However, when the image dimension is low relative to the number of sample images, such as the case in this thesis, calculating Σ is quite simple and practical.

3.3.2 Modeling the Face Class in Eigenspace

By finding a representation for the face class which reduces the dimensionality using PCA for example, not only does the complexity of feature-based algorithms improve in the modified image space, but the data is hopefully more succinctly represented in the feature space. This raises two questions: how do the eigenvalue magnitudes

decrease with increasing eigenvalue number, and consequently, how many dimensions are required to represent the entire face space?

The question of dimensionality (i.e. how many eigenvalues to retain) for a given problem has no definite answer. Jollffe [28] proposes a few techniques; some of which are statistically based while others are ad-hoc rules-of-thumb. For instance a cumulative percentage of total variance is defined such that a fixed percentage can be chosen (say 80% or 90%). This technique is widely used in the literature [40], [36].

The percentage of total variance retained in the new manifold is related to the value k (representing the first k eigenvalues) as:

$$t_k = 100\frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{m}\lambda_i} \tag{3.4}$$

where m is the total number of eigenvalues. A new technique examined in this thesis incorporates the number of retained eigenvalues k in the classifier model itself by considering it as a model parameter. Therefore, (C, γ, k) are "optimally" found using the procedure described in section 2.4.2. The outcome of this method is discussed in chapter 5.

In response to the issues just posed, an eigenspace was constructed from a set of 9716 face images used in this thesis [1]. Figure 3.3 shows a barplot of eigenvalue magnitude vs. eigenvalue number for the first 60 eigenvalues. The plot illustrates how a large percentage of the variance is captured in the first few eigenvalues; certainly a result of the high correlation of image-based facial features among the face images in the data set.

The correlation among face images in the training set definitely relies on the relative

view alignment of each image before PCA is applied. For instance, when the frontal-view face images are tightly cropped around the face showing only the main facial features (i.e. eyes, nose, mouth) such that the left eye is near the top-left pixel and at roughly the same scale (as is the case of this training set), regardless of facial variation among humans, the main *structure* (or template) of the facial features will usually coincide in each image within a few pixels. Obviously, the facial components of each human are unique and will also vary with illumination conditions giving different intensity values.

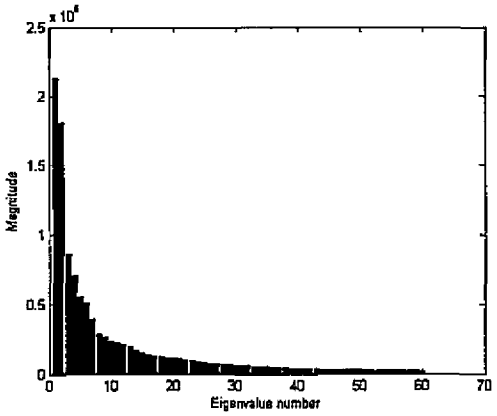


Figure 3.3: Magnitude of first 60 eigenvalues for face image set [1].

As an illustration of some of the conclusions above and properties of the eigenspace representation, Figure 3.4 demonstrates the relationship between variance and principal component number (or eigenvalue number) as well as the face class shape in the lower dimensional manifold.

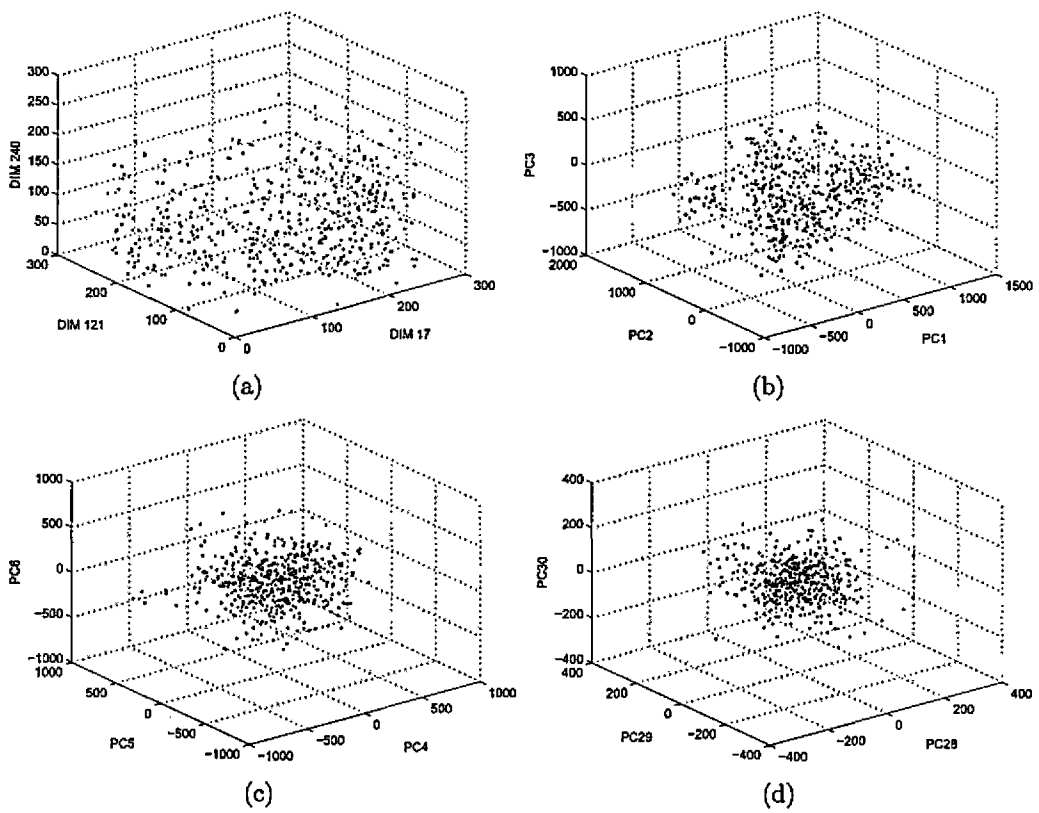


Figure 3.4: 30 dimensional eigenspace representation of a face image set [1]. 500 faces are plotted in each image showing 3 features. (a) Original raw data set illustrating randomness of 3 features (dimensions $17, 121, 240$). (b) First 3 dimensions of eigenspace representation. Notice the variance is largest in these dimensions (c) Dimensions $4, 5$ and 6 of the eigenspace representation. (d) Lowest three dimensions of the eigenspace representation.

3.3.3 Limitations of Principal Component Analysis

The effectiveness of any model is limited by the assumptions that it makes. This fact holds true with PCA. A few limitations of the PCA model are summarized [28]:

- PCA finds feature combinations that model the variance of a data set. However, these may not be the ideal features that separate the classes. Since PCA ignores class labels and only models the *variance* of a data set, its usage as a feature extraction technique may work poorly for pattern classification problems. However, in the research performed here with face image data, PCA showed improvements of 1-2% true positive rate when using PCA features as opposed to raw image-based features.
- PCA inherently makes the assumption that a distribution can be modeled as a Gaussian density function [28].
- The transformation $T(\mathbf{x})$ is linear. Therefore any nonlinear correlation amongst the data set is not captured by PCA. Face images in general are not exact linear combinations of a small orthogonal basis set. If they were, averaging two face images would result in a third face. However, this is not usually true because of varying facial structure and misalignment of facial features due to pose variations.

Despite its limitations, PCA provides a valuable tool in dimensionality reduction and has found widespread use in computer vision and many other fields.

3.4 Improving Face Class Modeling with Image Preprocessing

Appearance-based face modeling operates directly on the image intensities as features and therefore is inherently sensitive to lighting variations and different imaging sensor response curves. Two very common image processing techniques are used in this thesis to obviate the drawbacks of appearance-based modeling.

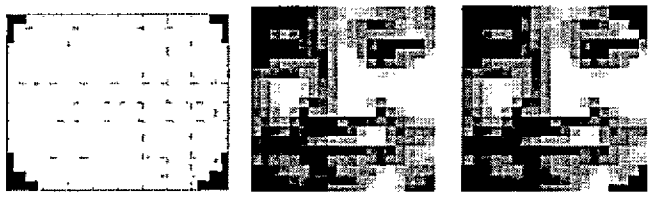
3.4.1 Spatial Filtering

A binary mask is used to eliminate some of the pixels around the edges of face image patterns. This is permissable without significant loss in face data since some of the border pixels are part of the background, or contain hair.

The main purpose of eliminating a few border pixels is to reduce unnecessary noise from the modelling, and consequently, the training process. The masking operation also has the added effect of reducing the dimensionality of the data space (in this case, from 361 down to 339). Figure 3.5 illustrates the masking operation used in this thesis.

3.4.2 Histogram Equalization

To help account for imaging sensor and lighting variations on the captured face images, global histogram equalization [18] is extensively used as a preprocessing step.



(a) (b) (c)

Figure 3.5: (a) Quasi-elliptical binary mask operator used. (b) Example face image. (c) Masked face image.

The transformation T is created such that:

$$\begin{aligned} s_k = \; & T(r_k) = \sum_{i=0}^{k} p_r(r_i) \\ = \; & \sum_{i=0}^{k} \frac{n_i}{n}, \quad k = 0, 1, 2, \ldots, 255 \end{aligned} \tag{3.5}$$

where n is the number of pixels in the image, n_i is the number of pixels of intensity value i, and p_r is the probability of pixel i in the image. Each pixel with intensity level r_i in the input image is then mapped to level s_k in the new histogram equalized image. Figure 3.6 illustrates the effect of histogram equalization from a small subset of the face image data set.

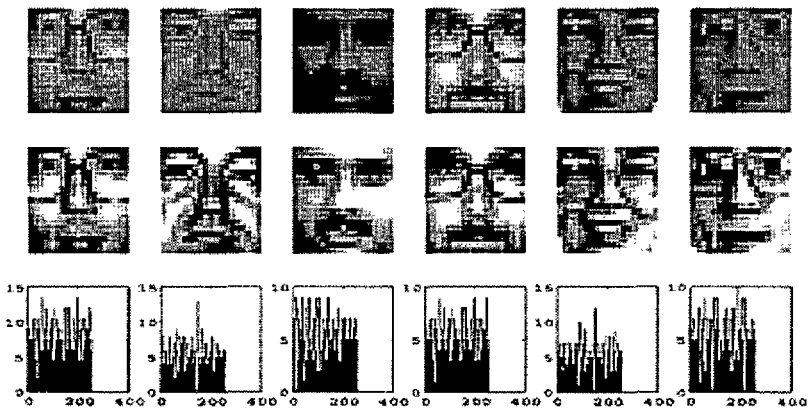


Figure 3.6: Effect of global histogram equalization on face images. (top) Original face images. (middle) Histogram equalized face images. (bottom) Corresponding histogram of resulting images.

3.5 Other Feature Generation Methods

3.5.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) [14] is another common technique for dimensionality reduction whereby a linear transformation is found such that the new data is maximally linearly separable. Contrary to PCA, LDA uses class labels to find the transformation. The theory behind LDA is correspondingly similar to the theory developed for Fisher's linear discriminant in Chapter 2.

To demonstrate the discriminating property of the LDA transform, Figure 3.7 shows a simple toy example where LDA can outperform PCA in the context of 1-D feature extraction for pattern classification of a 2-D data set. Using PCA, a basis $\mathbf{W_{PCA}}$ is found along the axis of maximum variance. However, as shown in the image, the projection of points onto this axis does not help separate the two classes, whereas

$\mathbf{W_{FLD}}$ does.

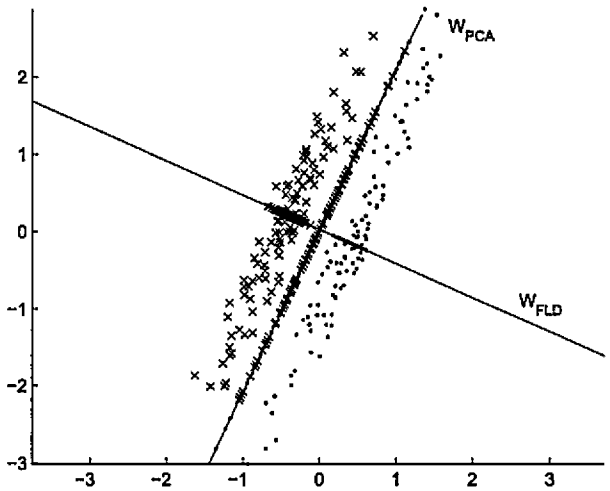


Figure 3.7: Synthetic example illustrating the class discriminating performance of LDA feature extraction. By projecting the points on $\mathbf{W_{FLD}}$ rather than $\mathbf{W_{PCA}}$, the classes are linearly separable.

In this thesis, PCA is used predominantly as a feature generator. Initial experimental results for face detection accuracy demonstrated a negligible improvement ($< 1\%$) in true positive rate from PCA features over LDA features.

3.5.2 Sub-sampled Images

Sub-sampling of the data (i.e. low pass filtering) is another technique for dimensionality reduction which produces a lower resolution representation of the original images. The images can be bilinearly interpolated to compute resulting values in the sub-sampled images. Sub-sampling was not used in the experiments of this thesis. However, Conde et al. [9] argue that sub-sampled raw images perform as well as PCA transformed images in a face recognition application.

3.5.3 Kernel PCA and LDA

Kernel PCA (KPCA) and LDA (KFLD) extend PCA and LDA to nonlinear manifolds to overcome the limitation of PCA and LDA as linear transforms. The input training vectors $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{m}$, $\mathbf{x}_i \in \mathcal{R}^N$ are mapped by $\Phi : \mathcal{X} \rightarrow \mathcal{F}$ to a high dimensional feature space $\mathcal{F}$. The linear PCA and LDA are applied to the mapped data $\{\Phi(\mathbf{x}_i)\}$.

In the case of KPCA, the computation of the principal components and the projection matrix $\mathbf{U}$ can be expressed similar to the method of section 3.3.1 in terms of dot products for which the kernel function $K(\mathbf{x}, \mathbf{x}')$ can be used. The projection matrix $\mathbf{U}$ is found such that the MSE of reconstruction:

$$\varepsilon_{KMSE}(\mathbf{U}) = \frac{1}{j} \sum_{i=1}^{j} \|\Phi(\mathbf{x}_i) - \widetilde{\Phi}(\mathbf{x}_i)\|^2 \tag{3.6}$$

is minimized.

Kernels can be used in the same context as described for SVMs in chapter 2. Gupta et al. [19] compared PCA, FLD, KPCA and KFLD using a SVM classifier for face recognition. They conclude that although kernels enable the use of a high dimensional feature space, they do not always ensure better performance than a linear subspace (such as PCA). Introduction of kernels might even degrade the overall performance if the classes are close to linearly separable. In their experiments, LDA/KFLD outperform PCA/KPCA for face recognition with KFLD giving the best overall performance.

Experimentation with KPCA and KFLD features was not performed in this thesis since additional kernel calculations are necessary for feature extraction, creating an additional computational overhead to the system. A good thorough description of kernel learning methods including KPCA, KFLD and SVM is given in [45].

3.6 Chapter Summary

Chapter 3 has presented an overview of appearance-based modeling of the face class. Some of the techniques for feature extraction and face preprocessing have been discussed. Intuitively, a more representative model of the face class will make the job of inductive learning, and ultimately, classification easier.

Experimental results employing theory developed in this chapter will be shown later in chapter 5.

Chapter 4

Selective Attention Methods for Locating Candidate Face Regions

Two conceptually different techniques for selective attention as they pertain to the thesis are discussed in this chapter. First, a method for selective attention from motion based on spatio-temporal differencing is described. Then, drawing on inspiration from biologically-based visual attention, a computational model for visual attention is described.

4.1 Visual Attentive Cues

An effective machine vision system must inherently possess the capability to focus valuable information processing resources to the most *important* components of the given perceptual input. In the context of face detection under practical unconstrained

environments, focused attentive cues can be as critical for success as the accuracy of the classifier itself.

Attentive cues are conveyed through the use of a *saliency map*[1]. The term "saliency map" originates from the work by Koch and Ullman [29] in modeling selective attention of the primate visual system. The term "binary saliency map" is used in this thesis to refer to a binary image which has the same image dimensions as the input. The map has values of "1" at locations which are considered "salient" or candidate regions to further explore. All other locations contain "0" values. Therefore, the saliency map is a means of selective attention by limiting the potential face regions to only those at, or in some constrained region, around the salient locations. The generalized saliency map contains non-zero values indicating the degree of saliency with respect to other locations in the map, and zero values otherwise. A set of "targets"[2] are then derived from the saliency map for a higher level process (such as a classifier) to inspect.

Motion and color are the two most obvious visual cues that can be valuable in machine vision systems. Color offers several benefits as a visual attentive cue, including robustness under occlusions and invariance to image translations. In the context of a face detection system, skin color cues can limit search regions to areas which might correspond to human faces. One major difficulty with color is that lighting variation can affect color models and may not be completely reliable under general conditions. Nevertheless, in practical systems, color has demonstrated effective results in face detection systems [23],[8].

In practical settings where only grayscale imaging sensors are available, motion is

[1]The terms "saliency map" and "selective attention map" are used interchangeably

[2]In this thesis, a "target" refers to a 2D coordinate point (x, y) representing the upper left pixel in a 19x19 window.

useful for inferring *active* regions of the visual input. In the context of a face detection system, motion is a meaningful cue since humans can usually be perceived as dynamic objects.

4.1.1 Motion

4.1.1.1 Spatio-temporal Differencing

The class of motion detection methods for selective attention construct a saliency map by segmenting images into two regions: *changed* and *unchanged* regions. To work in practical settings, static camera and constant brightness model assumptions are required for effectiveness.

In essence, spatio-temporal differencing follows in the spirit of image based approaches to feature extraction (PCA) from chapter 3. Changes in the intensity of image pixels are caused by motion, imaging noise and varying illumination. Since motion is embedded in this temporal change, it cannot be determined directly. However, an estimate of motion can be recovered under constant illumination assumptions and negligible imaging sensor noise.

Localized intensity changes at each image coordinate $I(x, y)$ provide significant information about the motion of a scene being imaged. The most common approximation to motion change uses simple pixel-wise subtraction of two successive frames in time, i.e.

$$\frac{\partial I(x, y, t_k)}{\partial t} \approx I(x, y, t_k) - I(x, y, t_{k-1}), k = 1, 2, ... \tag{4.1}$$

where t_k is the time at frame k.

This simple yet widely used method has found application in many human motion tracking applications. See [35] for a survey on human motion capture.

With this simplicity come limitations. Changes in ambient and directional illumination or reflection can add error (noise). A common method for reducing sensitivity to intensity change caused by factors other than motion is to preprocess the images with a smoothing function before performing frame differencing (FD), giving:

$$\frac{\partial I(x,y,t_k)}{\partial t} \approx \frac{\partial}{\partial t}(G(x,y,t_k) \otimes I(x,y,t_k)), k = 1,2,3,\cdots \tag{4.2}$$

where $G(x,y,t_k) = \sqrt{2\pi}\sigma A e^{-2\pi\sigma^2(x^2+y^2)}$ is a standard spatial domain representation of the Gaussian low pass filter.

Then, a binary saliency map $FD_{t_k}(x,y)$ at time t_k can be constructed indicating coordinates of motion as "1" and static locations as "0". In order to distinguish non-zero differences in equation (4.1) that are noise from those that are due to local motion, thresholding is used such that:

$$FD_{t_k}(x,y) = \begin{cases} 1 & \text{if } |I(x,y,t_k) - I(x,y,t_{k-1})| > T \\ 0 & \text{o/w} \end{cases}, k = 1,2,3,... \tag{4.3}$$

given an appropriate user defined threshold T.

The simple method of equation (4.3) gives a crude approximation to motion in an image but is ineffective in difficult complex video scenes used in the experiments performed in chapter 5. Figure 4.1 uses the newscaster [2] sequence to demonstrate how a dominant, complex background can divert attention away from the anticipated regions of interest (i.e. newscasters) by adding noisy unwanted attention regions in the map.

(a) (b) (c)

Figure 4.1: Simple demonstration of temporal differencing. (a) Frame 1 of the newscaster sequence [2]. (b) Image differencing of frames 1,2 with T=2. (c) Image differencing of frames 1,2 with T=0.

4.1.1.2 Temporal Integration

The abrupt unwanted changes between successive frames as described above can be smoothed out to some extent by integrating memory into the detection algorithm; ensuring improved spatial and temporal continuity. The frame difference with memory (or $FDM_k(x,y)$) attention map is obtained by differencing the present frame and a weighted sum of the past k frames before doing binary thresholding as before:

$$FDM_{t_k}(x,y) = \begin{cases} 1 & \text{if } |I(x,y,t_k) - \hat{I}(x,y,t_{k-1})| > T \\ 0 & \text{o/w} \end{cases} , k = 1,2,3,... \qquad (4.4)$$

where $\hat{I}(x,y,t_{k-1}) = (1-\beta)I(x,y,t_k) - \beta\hat{I}(x,y,t_{k-1}), k = 1,2,...$

and constant $0 < \beta < 1$ influences the aggregate contribution of past frames into the image differencing. By integrating more history into the frame differencing technique, the FDM method increases the likelihood of eliminating false motion indications [18]. Note that a non-causal system such as a pre-stored video sequence is required for

potential real-time use. Otherwise, a delay of k frames is required to compute the attention map.

4.1.2 Saliency Map Creation from Motion

The saliency map generated by temporal differencing from equation (4.4) leads to a set of regions in the image of local motion. In the context of this research, the goal of the saliency map is to define a set of binary labels indicating candidate locations for faces. Therefore, eliminating isolated regions of motion with no potential face is desired.

Two ad-hoc approaches to saliency map creation from motion cues are considered:

Approach 1: Compute $FDM_{t_k}(x, y)$ with $k = 2$, $\beta = 0.5$ using a low threshold $T = 1$, thereby making the system highly sensitive to motion. Then, using a 16-connected neighborhood filter, small isolated patches of motion are removed. Therefore, ideally, only larger clusters of points remain in the saliency map and small isolated patches which assumingly do not contain facial motion, are removed.

Approach 2 Compute $FDM_{t_k}(x, y)$ with $k = 20$, $\beta = 0.9$, using a large threshold $T = 65$. The intention is that by using a larger threshold and incorporating much more knowledge of the past frames, the saliency map reduces to only a small number of *highly active* motion regions consistent over the past 20 frames.

From the results of a few standard video sequences containing faces, the second approach showed better results by producing less erroneous targets and more points

concentrated around known faces in the sequences. Figure 4.2 illustrates the results of approach 2 using the newscaster, irene and akiyo video sequences [2].

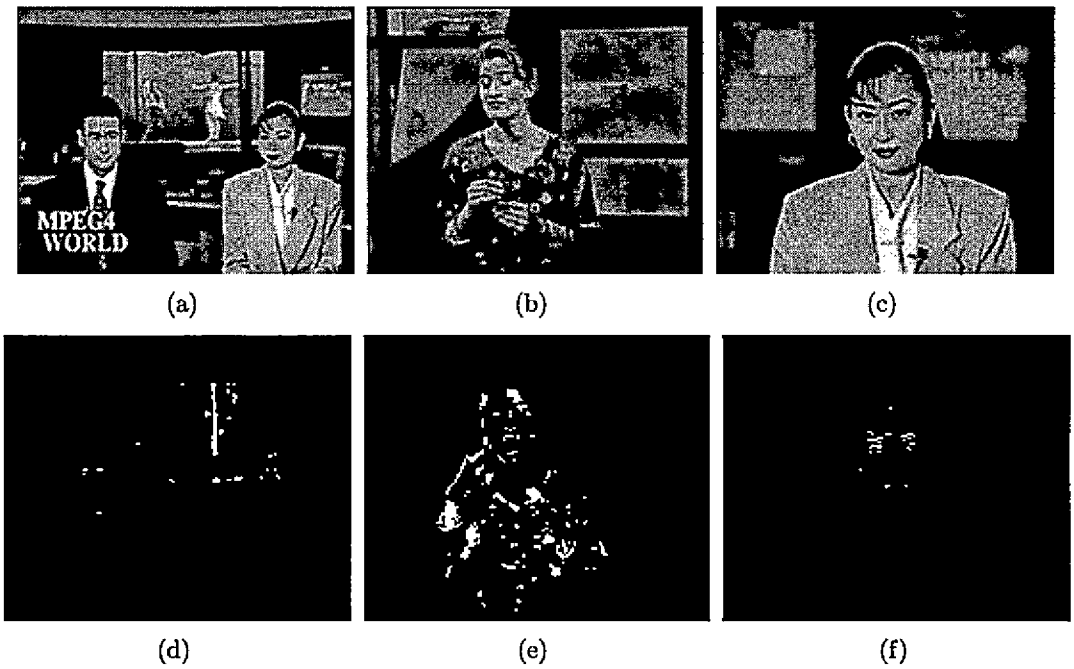


Figure 4.2: Illustration of approach 2. (a) Frame 1 of newscaster sequence. (b) Frame 1 of irene sequence. (c) Frame 1 of akiyo sequence. (d) newscaster saliency map, $\beta = 95$, $k = 20$, $T = 65$, pts=129. (e) irene saliency map, $\beta = 95$, $k = 20$, $T = 65$, pts=625. (f) akiyo saliency map, $\beta = 95$, $k = 20$, $T = 65$, pts=48.

Since faces are assumed to be constrained to nearly frontal upright poses in images, one additional improvement is added to the selective attention model to eliminate spurious face regions. Once the saliency map is constructed, the saliency points (i.e. (x, y) locations with a value of "1") are examined in raster-scan fashion (left-to-right, top-to-bottom) by the classifier such as in Figure 4.3. If motion of a human is present in the saliency map, faces will generally be processed before other parts of the body. In situations where humans are not arranged in a frontal upright pose, this heuristic

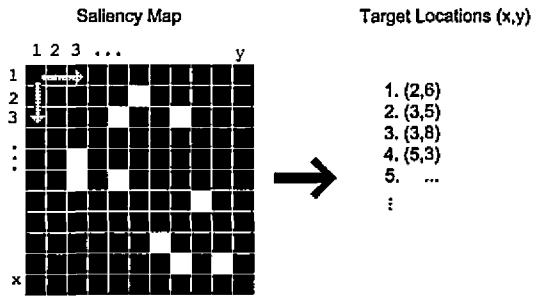


Figure 4.3: Simple example of binary saliency map. Targets are read in a left-to-right, top-to-bottom form

is not a valid form of speed enhancement, yet obviously does not perform any worse than all other scan orderings since all saliency points are eventually processed.

4.2 A Biologically-inspired Model for Selective Attention

Although motion provides a simple visual cue for focusing attention, it is not without limitations. The strong assumptions of constant brightness and static camera limit the effectiveness of simple image differencing. As well, scenes with complex background motion such as in Figure 4.1 can drastically enlarge the number of saliency targets, thereby increasing the complexity of face detection. Additionally, creating a saliency map for a static grayscale image is not possible.

A second fundamentally different approach to selective attention is discussed in this

section. The selective attention model developed by Itti [26] is built upon a computational model of bottom-up[3] visual attention in primates.

The purpose of introducing this selective attention model is to provide some insight into why such a model can be useful for simplifying the input from highly complex visual scenes. In the primate visual system, it is understood that only select locations of a scene are processed in detail; referred to as the salient fixations of a scene. Remarkably, rather than attempt to interpret visual scenes in a parallel fashion, biological systems employ a serial search strategy using two guiding influences: bottom-up (image based) cues and top-down (task dependent) cues [25]. Research suggests that the bottom-up guidance of visual attention in primates relies on simple cues such as local intensity, color and orientation contrasts [29]. It is surprising, from a machine vision point-of-view, that with the absence of complex visual cues involving rotation and scale invariant template matching mechanisms or other algorithms, the performance of selective attention in a biological system works very well at all. Because of the variety of sophisticated algorithms invented to solve computer vision problems, it is counterintuitive to believe that simple filtering from color, orientation and intensity cues can provide enough information to make conclusions about the most important locations in an image.

4.2.1 Model

Itti [26] proposed a computational model of selective attention derived from the bottom-up visual attention mechanism of biological systems. The model is based

[3]Cognitive scientists believe that humans use information gathered through the senses (referred to as bottom-up processing) and information stored in memory (often called top-down processing) in a dynamic process to construct meaning about their environment and their relations to it [24].

on the so-called "feature integration theory" of human visual search strategies [26]. That is, a set of low level features are combined ultimately into a master saliency map.

Figure 4.4 shows the general architecture of the model. Based on the input provided from a static color image, a six level Gaussian pyramid of sub-sampled images of the input is constructed. A set of visual features are extracted from each level into *channels*. Typically the low-level features are contained in seven channels including intensity contrast, red/green and blue/yellow color opponensies, and 0^o,45^o,90^o, and 135^o orientation contrasts giving a total of 42 feature maps.

The opponency and contrast operations are formed through a "center-surround" operation which has a biological basis in the primary visual cortex [31]. This is implemented as the difference between fine c and coarse s scales (two consecutive images of the pyramid) denoted "$\ominus$". The across-scale difference is achieved using pixel-by-pixel subtraction from the interpolated lower resolution image. The center-surround operation is sensitive to local spatial discontinuities (i.e. edges) and is therefore useful for detecting locations which stand out from their local surrounding pixels.

From the intensity channel $I = (r + g + b)/3$ where r,g and b are the red, green and blue channels of the input image, one set of feature maps are constructed using the intensity channels computed as:

$$I(c, s) = |I(c) \ominus I(s)| \tag{4.5}$$

The color channels are normalized by I in order to separate hue and intensity such that $R = r - (g + b)/2$, $G = g - (r + b)/2$, $B = b - (r + g)/2$ and $Y = (r + g) - |r -$

$g|/2 - b$ for red, green,blue and yellow respectively. A second set of maps are created from the color channels producing a set of "color opponencies". Such chromatic opponencies exist as pairs in the primary human visual cortex for the red/green, green/red, blue/yellow and yellow/blue pairs [25]. The model computes these using the center-surround operation as:

$$\begin{aligned} RG(c,s) &= |(R(c) - G(c)) \ominus (G(s) - R(s))| \\ BY(c,s) &= |(B(c) - Y(c)) \ominus (Y(s) - B(s))| \end{aligned} \tag{4.6}$$

Local orientation feature information $O(\theta)$ where $\theta \in \{0^0, 45^0, 90^0, 135^0\}$ is extracted from I using oriented Gabor filters which approximate the receptive field impulse response or orientation-selective neurons in the primary visual cortex [25]. Orientation contrast feature maps are calculated using the center-surround operation as:

$$O(c,s,\theta) = |O(c,\theta) \ominus O(s,\theta)| \tag{4.7}$$

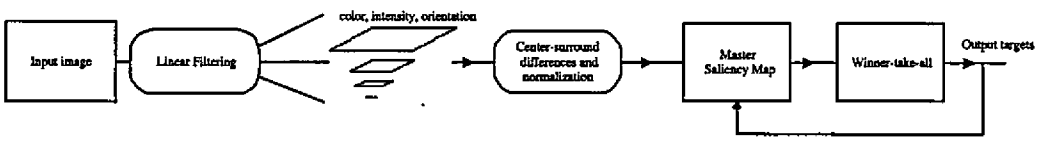


Figure 4.4: Selective attention model architecture.

Once the set of 42 feature maps are constructed, they are normalized and linearly combined into a master saliency map. Depending on the application, the weighting of each channel in the linear combination stage can be adjusted to favor individual features. For instance, in the experiments performed on grayscale images containing faces, the color channels are given a weighting of zero, while the intensity and orientation contrast channels are combined with equal weighting.

The normalization step is a critical step in the model since it attempts to combine feature maps of incomparable modalities (contrast, orientation, color) which have very different dynamic ranges and extraction mechanisms. For instance, how does the saliency of a sharp 45° discontinuity at one location compare to a strong change in intensity at another? And since 42 feature maps are being combined, a salient object appearing in one feature map may be shrouded by less salient or even noisy objects present in some of the other feature maps. Normalization is therefore achieved by promoting feature maps with a small number of strong peaks of activity while suppressing maps containing numerous comparable peak responses.

Once all the features have been combined into a master saliency map, focus of attention could be directed to regions of the largest saliency map values, in order of decreasing values. However, a biologically-based competitive neural network (winner-take-all) [29] is used to extract target locations in a serial fashion. The details and rationale for the target extraction mechanism are given [25], [29] as a starting location.

4.2.2 Incorporation into a Face Detection Framework

A largely experimental basis for incorporating the biologically-based selective attention model into a face detection system is given in this thesis. In a face detection context, the principal assumption made is that human faces constitute salient objects. In fact, experiments in chapter 5 on images containing faces (including the CMU face test set [38]) empirically demonstrate this notion. It is very important to note that the face/non-face classifier is not integrated into the overall face detection system at this point; the selective attention mechanism is merely a front-end system for the overall face detection framework. Experimental results of the overall face detection

system are described later in chapter 5.

Figure 4.5 illustrates an example of the selective attention model applied to two sample images from the CMU test set [43].

Figure 4.5: sample images from CMU test set after being processed by the front-end selective attention system.

Only the first few target coordinates are shown. The resulting targets are overlaid on the images as large circles, with arrows indicating sequence of targets acquired. Note that in the right image the set of face targets is not exhaustive and does not pick out all faces in the image in the first 7 targets. In contrast to the right image, clearly as more faces are present in an image, the less salient each individual face becomes since they are competing for visual attention.

One of the principal advantages of such a model is that since it uses low level features including intensity, orientation and color, the selective attention mechanism is invariant to high level object description, making it extendable to a potentially wide range of computer vision applications. In other words, saliency is defined at a very primitive level such that faces, vehicles, animals, or any other objects all fall into the capable realm of this selective attention model.

In chapter 5, the two selective attention mechanisms discussed in this chapter will be used as a front-end processor for a support vector machine-based face detection system.

Chapter 5

System Evaluation

Experimental results of a novel face detection system are described in this chapter. An overview of the system is presented as a client-server architecture and a description of each component is described. The system is evaluated using a set of common image stills and video sequences to demonstrate its effectiveness.

5.1 System Overview

A system for frontal-view face detection is developed from the concepts of previous chapters. Given an image under examination, a front-end selective attention system uses low-level feature analysis to derive a small set of image coordinate targets which are then passed on to the classifier for higher-level inspection. An overview of the system is given in Figure 5.1 and encompasses all of the concepts described in chapters 2-4.

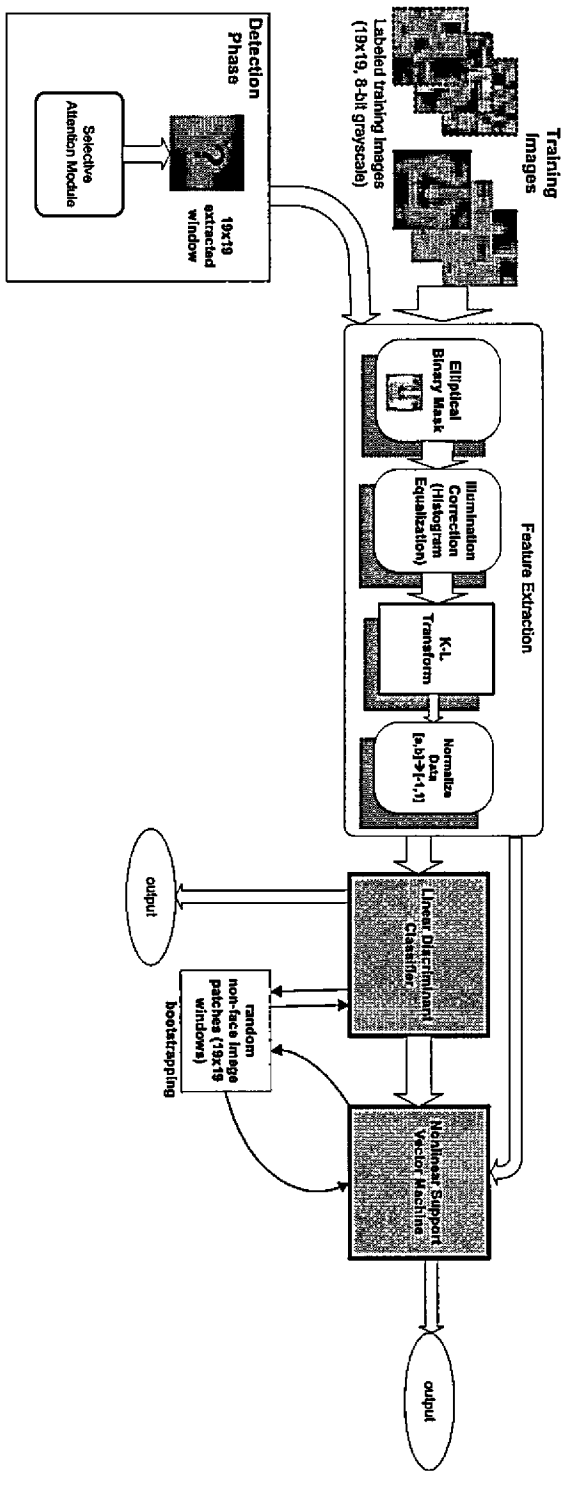


Figure 5.1: System block diagram overview of face detection system.

5.1.1 Client-Server Arrangement

The two major components of the face detection architecture are organized in a client-server arrangement as illustrated in Figure 5.2. The front-end selective attention module acts as the server listening for and accepting TCP/IP connections (in this case, on port 50000). Main operation of the face detection system is controlled through the client which acquires an image to be processed (either externally through a camera, or a previously stored image or video sequence).

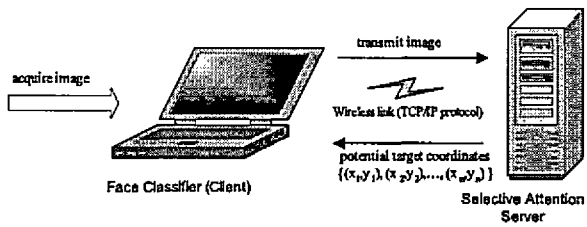


Figure 5.2: Client-server arrangement for face detection system.

Once the image is acquired, the client then connects to the server on port 50000 and transmits the image. The selective attention server then begins processing the image and responds to the client in a serial fashion with a set of points (x, y) as 8-bit unsigned integer values indicating target locations for the classifier to inspect. The classifier and selective attention module can therefore operate independently and in parallel once the classifier has been given at least one target point. The receiver buffer on the client computer collects the target points as they are received, and grants the classifier continual access to the buffer in order to examine the received targets.

The client-server arrangement using TCP/IP socket programming facilitates communication across multiple platforms. In this case, the selective attention server is written in Python/C++ under Fedora Core Linux, while the client runs in Microsoft Windows XP. Additionally, since the physical connection layer is abstracted away

from the application layer, the system will operate over standard 802.3 for example, or wireless 802.11b in the case of this experiment.

5.1.2 Selective Attention Server

Two different selective attention mechanisms are used in this system. For grayscale image stills and video sequences, the selective attention system described in section 4.2 was invoked. Since no color information is present, the model assigns a weighting of zero for the color channel and assigns an equal weighting to the intensity and orientation channels.

To demonstrate the model's ability to locate correct face *regions*, the CMU test set [43] was applied to the selective attention system. The test set contains 130 images with one or more faces per image. A total of 507 non-occluded faces are present in the data set, with ground-truth face locations known.

In the first experiment, 10 targets were acquired per image with an average computation time of 2425ms for the 10th target. Table 5.1 summarizes the results for all 130 images as well as a subset (96 of the 130 images) containing only 1-4 faces in them. The results are also summarized for targets with a direct hit on a face region, and results for near hits within a region of 25 horizontal and vertical pixels. As expected, the model performs well at selecting targets when only a few faces are present in an image. With only the first 10 targets, nearly 78% of the face regions (within 25 pixels) are located.

Similar conclusions are drawn from the result of using 30 targets with an average computation time of 3132ms for the 30th target. Table 5.2 shows that with the first

30 targets, approximately 84% of all face regions are located (within 25 pixels).

CMU data set	on face			within 25 pixels		
	faces found	total faces	%	faces found	total faces	%
images containing only 1-4 faces (96)	83	127	65.3	99	127	77.9
entire data set (130)	148	507	29.2	192	507	37.8

Table 5.1: Selective attention results for CMU test set [43] using 10 attention shifts per image.

CMU data set	on face			within 25 pixels		
	faces found	total faces	%	faces found	total faces	%
images containing only 1-4 faces (96)	91	127	71.6	107	127	84.2
entire data set (130)	157	507	30.9	304	507	59.9

Table 5.2: Selective attention results for CMU test set [43] using 30 attention shifts per image.

Figure 5.3 shows some example images from the test set with the first 30 targets overlaid on the image.

A second experimental approach to selective attention on video sequences is compared to the biologically-inspired model. The approach described in section 4.1.2 selects targets by using a 20 frame history to compute the saliency map based on spatio-temporal differencing. Although the saliency map can be created very quick from only 20 frames, the number of target points can be much larger than the number generated by the biologically-inspired system. A comparison is done using both selective attention models. When using the biologically-inspired model, the first frame of the sequence is used to select the first 10 targets with an average computation time

Figure 5.3: sample images from CMU test set after being processed by the front-end selective attention system with 30 acquired targets. The targets are overlayed onto the images as black asterisk symbols.

of 1310ms on a Pentium 4-2.8 GHz machine. Experiments using a set of 6 standard 176x144 CIF video sequences are summarized in Table 5.3.

	Motion-based model		Biologically-inspired model	
Video Sequences	# targets acquired	Hit rate (within 25 pixels)	# targets acquired	Hit rate (within 25 pixels)
akiyo	48	1 / 1	10	1 / 1
irene	625	1 / 1	10	1 / 1
carphone	268	1 / 1	10	1 / 1
trevor	3147	5 / 6	10	4 / 6
newscaster	129	2 / 2	10	2 / 2
silent	917	1 / 1	10	2 / 2

Table 5.3: Results comparing motion-based and biologically-inspired selective attention.

The results demonstrate that motion as a visual cue can be effective when the most dynamic objects in a scene are the faces or humans. Otherwise, unwanted background noise can expand the number of targets to over 3000 locations of a possible 25344 in the 174x144 images.

5.1.3 Classification

A two-stage binary classifier as described in section 2.5 is used to process the candidate face regions and output a "1" or "-1" corresponding to face or non-face. The classifier operates on feature vectors of fixed size k, where k is ultimately determined by the dimension of the reduced space using PCA or LDA.

5.1.3.1 Image Preprocessing and Feature Extraction

As shown in Figure 5.1, a sequence of operations are first applied to the candidate window before being passed to the classifier. First, the image window is convolved with a quasi-elliptical binary mask as described in section 3.4.1, removing some of the pixels lying close to the boundary of the window pattern which introduce noise into the training process. The image space dimension (number of pixels) is reduced from the original 361 dimensions down to 339. Histogram equalization is then applied to each image compensating for variations in illumination brightness. Then, the face image space is reduced from 339 to k dimensions via either PCA or LDA. Finally, the data is scaled to the range $[-1, 1]$.

Commonly in the machine learning community, when using algorithms such as neural networks or SVMs, the attributes of the feature vectors are linearly scaled either to the range $[-1, 1]$ or $[0, 1]$. While the linear scaling process might be destructive if the probability density function of the feature vectors are highly nonlinear, it can be advantageous in prohibiting attributes in greater numeric ranges that could dominate those in smaller numeric ranges.

A feature vector $\mathbf{x}$ is rescaled to the range $[-1, 1]$ using the transformation $\mathbf{x}_n = 2(\mathbf{x} - min_{\mathbf{x}})/(max_{\mathbf{x}} - min_{\mathbf{x}}) - 1$. The vectors $min_{\mathbf{x}}$ and $max_{\mathbf{x}}$ are calculated from the original training data and must be stored for rescaling future testing feature vectors. A subtle yet important point to note is made in the following example. If the first attribute of the *training* data was scaled from [-10,10] to [-1,1], and the first attribute of the *testing* data lies in the range [-12,7], it is incorrect to rescale this data to [-1,1] but rather should be scaled to [-1.2,0.7].

5.1.3.2 Training

The two-stage classifer is trained using a test set of images. In the research presented here, a set of very low-resolution "thumbnail" images are used in which only the main frontal facial region is visible. One of the advantages of using this reduced facial image structure is to minimize the effects of hair styles that change in appearance easily and frequently. Secondly, this study aims to investigate face detection performance using very minimal facial information. In this case, the images are 19x19 = 361 pixel 8-bit grayscale images.

The training data set used in these experiments was built from the MIT-CBCL frontal face database [1] consisting of over 400 male and female subjects cropped and centered in the image while posing under varying lighting conditions and facial expressions. The image set contains a broad range of facial conditions including variations in skin color, glasses, and facial hair. Figure 5.4 shows a few sample images from the dataset. To improve generalization of the classifier for handling minor variations in frontal and upright pose, each face was transformed via small reflections, scaling, rotations and translations, expanding the final training set to 9716 19x19 grayscale face images.

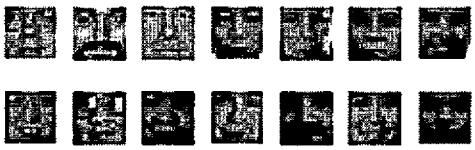


Figure 5.4: Sample images from the MIT-CBCL training data set

The training data is preprocessed as described in section 5.1.3.1 before being passed into the two-stage classifier. Once the classifier is initially trained, a bootstrapping procedure (see section 2.4.1 for an introduction) inspired from the work of Osuna et al. [38] and Sung and Poggio [47] is used to improve classification results by

appending to the training set important false positive patterns (near-faces) that lie close to the face/non-face boundary. A concise summary of the two-stage classifier training process is given in Table 5.4:

1. Train FLD classifier $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ using an initial training set $\mathcal{D}$ of 9716 preprocessed faces and 10000 preprocessed non-faces.
2. Test FLD classifier on a set of 472 new preprocessed faces and 1000 new preprocessed non-faces as well as the initial training set $\mathcal{D}$.
3. Collect all ambiguous and erroneous patterns (FP, FN) from (2) into a new set $\mathcal{S}_{svm}$.
4. Train the second-stage SVM classifier on data set $\mathcal{S}_{svm}$. The SVM parameters (C, γ) are found using 3-fold cross-validation (see section 2.4.2).
5. **Bootstrapping procedure:**
   (a) Test the first-stage FLD classifier with 10000 preprocessed non-faces. Collect ambiguous and erroneous (i.e. FP) patterns into set $\mathcal{B}_{fld}$.
   (b) Test second-stage SVM classifier with patterns $\mathcal{B}_{fld}$. Collect any errors made (i.e. FP) into new set $\mathcal{B}_{svm}$.
   (c) Retrain FLD classifier with training set $\mathcal{D} \bigcup \mathcal{B}_{fld}$.
   (d) Retrain SVM classifier with training set $\mathcal{S}_{svm} \bigcup \mathcal{B}_{svm}$.
   (e) repeat (a) if desired.

Table 5.4: Two-stage classifier training procedure.

In step 1 of Table 5.4, the FLD classifier requires the user defined parameters T_a and T_b which reflect the speed/accuracy tradeoff of the system (see eq. (2.22)). A good indicator for a threshold value T_a and T_b can be based roughly on the margin value $1/||\mathbf{w}||$ from a single SVM classifier trained on the same data. From the experiments

performed, most values were in the range $0.0002 - 0.0004$.

As described in section 3.3.2, the chosen number of dimensions k to retain is user defined. One consideration is to include k into the list of optimization parameters in eq. (2.20) such that:

$$(C^*, \gamma^*, k^*) = \max_{C,\gamma,k} CVA \tag{5.1}$$

To measure the relationship between the number of support vectors, the threshold value $T = -T_a = T_b$ and the feature vector dimensionality k, a plot is constructed in Figure 5.5 based on the original training data [1]. As the number of retained eigenvalues k increases, the number of SVs also increase since the SVM is requiring more SVs to model additional noise in the system supplemented by the added eigenvalues. The plot also shows nearly a linear relationship between threshold T and the number of SVs.

5.2 Multiscale Searching

Once a set of targets are passed to the classifier, a searching strategy is used to investigate a small region surrounding the target[1]. A limited range sliding window (25 pixels horizontal and vertical) at multiple scales is applied to each received target point. Obviously, since the classifier can only handle fixed input windows sizes of 19x19, a technique is required to examine larger windows areas in an image.

A pyramid of images is created similar to [43] by repeatedly subsampling the window

[1]The convention used here is that a target corresponds to the pixel in the upper left corner $(0,0)$ of the window

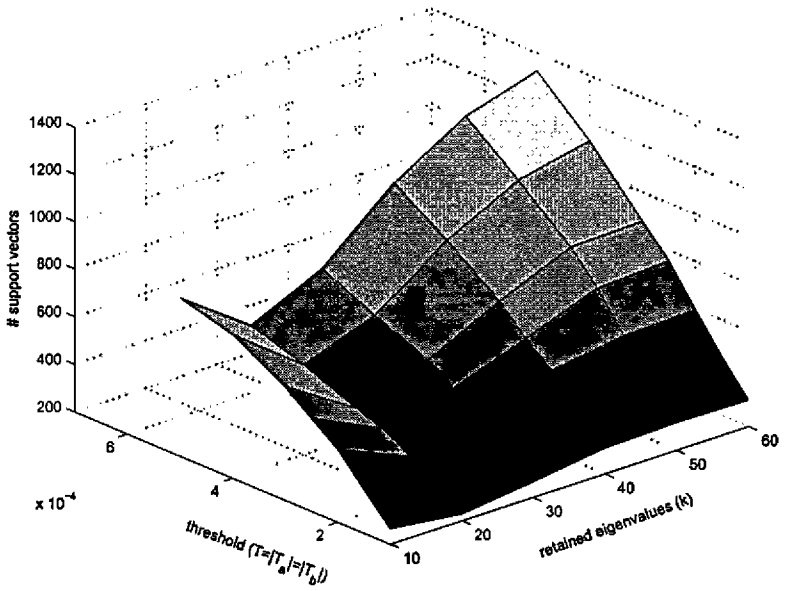


Figure 5.5: Relationship between number of support vectors, threshold value and input space dimension based on the initial training data.

by approximately 25% creating 10 levels from 19x19 up to 190x190 in size (see Figure 5.6. The windows sizes are 19x19, 24x24, 30x30, 38x38, 47x47, 58x58, 73x73, 90x90, 130x130 and 190x190. Each window is bilinearly interpolated down to the original training size of 19x19 pixels for further preprocessing and examination by the classifier. Consequently, the system can detect faces as small as 19x19 and as large as 190x190.

5.2.1 Improving Results with Overlap Detection

Figure 5.7 shows ROC curves for the two-stage classifier and the single SVM classifier based on 472 19x19 face images and 20000 19x19 non-face images from the MIT-CBCL test set [1].

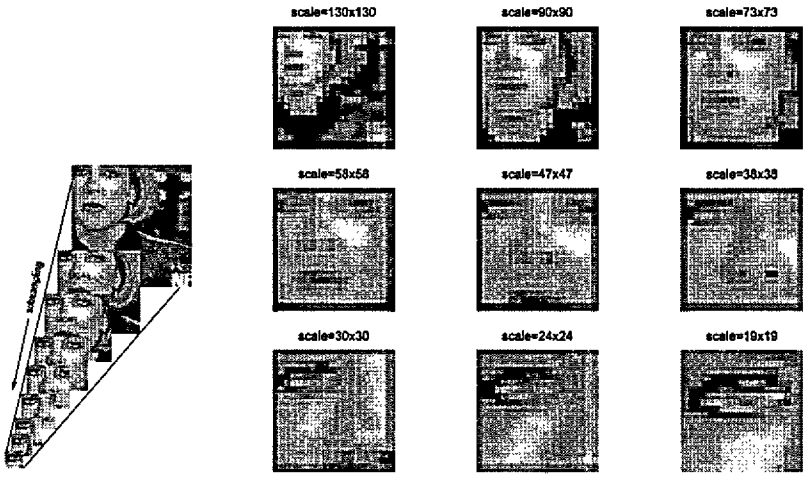


Figure 5.6: Multiscale searching.

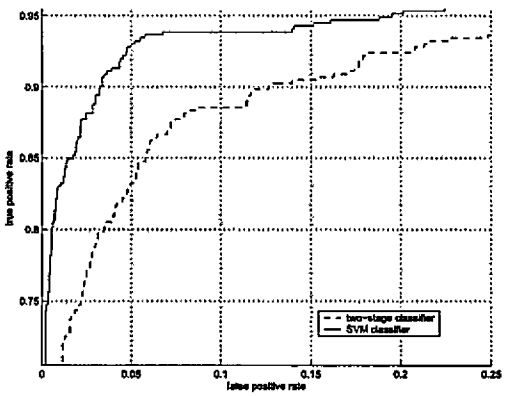


Figure 5.7: ROC curves for two-stage classifier and SVM classifier based on MIT-CBCL test set.

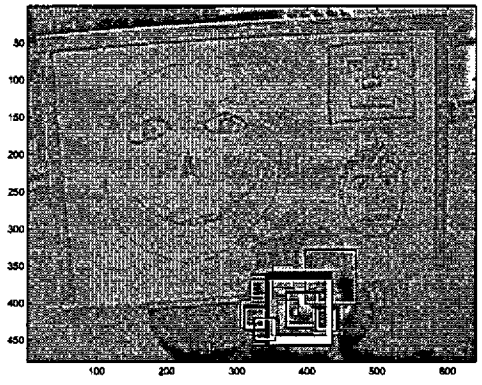


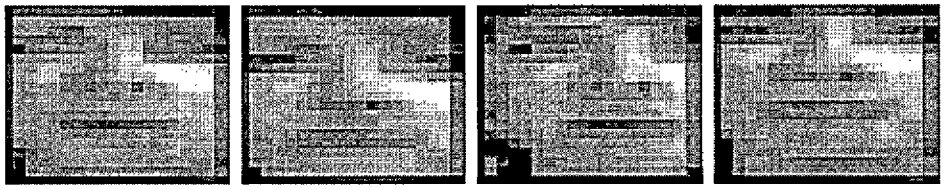
Figure 5.8: henry image from CMU data set processed without applying overlap detection. Boxes with thick black & white borders indicate multiple hits within a very small region, whereas boxes with thin borders indicate isolated detections.

As demonstrated in Figure 5.7, one possible point of operation on the curve using the two-stage classifier is approximately 85% true positive rate with a 5% false positive rate. The point (0.05,0.85) indicates that when given 100 face images and 100 non-face images, the classifier will recognize 85 of 100 faces while identifying 5 non-faces as being faces.

One of the major limitations of this arrangement for face detection is that the classifier assumes each window pattern is independent. However, there is an obvious *spatial* relationship between window patterns. Consider Figure 5.8 after being processed by the face detection system. Notice how the number of window detections is high around a true face, whereas isolated windows are spotted at other locations. This observation yields an apparent heuristic technique called *overlap detection*, that can be used to drastically reduce the FP rate, described first in [5] and used in many other published face detection systems.

As described, usually true faces will give correct classification in 2 or 3 consecutive scales and locations. To demonstrate this supposition, Figure 5.9a illustrates an

example window from the cnn2600 image of the CMU test set. A few minor horizontal and vertical translations of 7 pixels are applied to the images. All four images are detected as faces giving comparable classifier *scores*[2].



(a) reference image (b) (-7,0) translation (c) (0,-7) translation (d) (7,0) translation

Figure 5.9: cnn2600 test image from CMU test set. Minor translations are applied to the reference image. Resulting classifier scores are: (a) 0.2475 (b) 0.1831 (c) 0.0441 (d) 0.0988.

Due to the extensive face image set used for training, the classifier is relatively immune to minor variations in translation as well as pose and scale. This notion is used to reduce the number of false positives occurring in the image and confirm correct face regions. At each location and scale, a 3x3 neighborhood filter at neighboring scales (one scale higher and lower) is applied giving a count (maximum value of 27) of the number of detections in a neighborhood (cube). Finally, a user defined threshold value empirically chosen as 18 is used to remove isolated detections by only retaining detections with a large number of multiple hits within a 3x3 region and within 3 scales in the image pyramid. There are a few pathological cases where this heuristic can fail (such as the case of partially occluded faces [43]) but overall, as shown in the next section, the false positive rate is reduced by 10-100 times.

[2]A classifier's score can be thought of as a numerical value indicating the degree to which the pattern belongs to a particular class. The binary output of the classifier is the absolute value of this score.

5.3 Final Results

Figure 5.10 shows the final ROC curve for the two-stage SVM classifier compared with a single SVM classifier system. The results are based on the CMU test set, with user-defined threshold $T_a = -T_b = 0.015$, and $C^* = 2000$, $\gamma^* = 0.42$ and $k^* = 32$ found using eq. (2.20). The front-end biologically-inspired selective attention system was used to acquire 50 targets per image.

From the initial training set, 324 faces and 808 non-faces filtered through the intial (FLD) classification stage resulting in the SVM classifier producing 284 support vectors. Based on the CMU test set, the share of the classification work was divided as 44% for the first (FLD) stage and 56% for the second (SVM) stage. Note that when the SVM classifier was trained and used independently on the training data, between 800-2500 support vectors were present depending on system parameters.

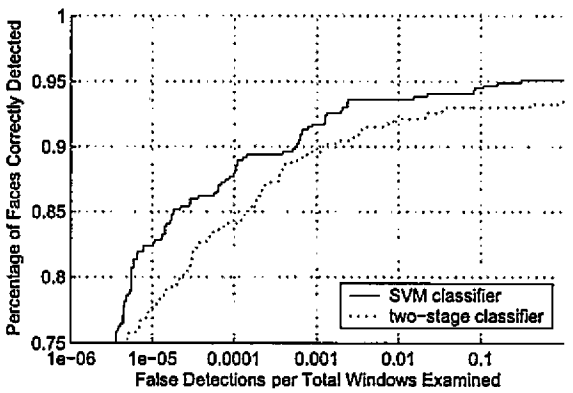


Figure 5.10: ROC curve comparing two-stage and SVM classifiers based on CMU test set, with user-defined threshold $T_a = -T_b = 0.015$, and $C^* = 2000$, $\gamma^* = 0.42$ and $k^* = 32$ found using eq. (2.20).

A detection rate of 80.6% with 28 false positives ($\sim 1.1\text{x}10^{-5}$ FP rate) was obtained from the experiment. Comparable detection rates are obtained by Rowley [43] citing

rates in the range of $77.9\% - 90.3\%$ with small FP rate ($< 10^{-4}$) based on the CMU test set. Osuna et al. [38] also give similar detection results based on the CMU test set. Table 5.5 gives a summary and comparison of reported detection rates from other common approaches.

Face Detector	Detection Rate (%)	# FP
Proposed system	80.6%	28
Rowley et al. [43]	86.2%	23
Viola and Jones [52]	85.2%	31
Garcia and Delakis [17]	90.3%	8

Table 5.5: Comparison of performance results for face detection systems.

A few sample images from the CMU test set processed by the face detection system are shown in Figure 5.11. From visual inspection, the face detection system demonstrates some capability to handle minor variations in yaw and tilt from the frontal view pose.

Figure 5.11: Sample images from CMU test set processed by face detection system.

Using 50 targets per image while searching within a 25 pixel radius at 10 scales,

only 187,500 19x19 windows per image are examined, giving a total of 24,375,000 windows for the entire CMU test set.

While the detection results are comparable to other published face detection systems, the added gain lies in the reduced complexity in searching and classification. Table 5.6 shows that whereas Rowley et al.'s [43] system examined 83,099,211 20x20 pixel windows for the CMU test set, only 24,375,000 19x19 windows are examined in this system.

	# Windows Examined	Detection Rate ($<$ 10^{-4} FP rate)
Proposed system	24,375,000	$74.2 - 88.7\%$
Rowley et al. [43]	83,099,211	$77.9 - 90.3\%$

Table 5.6: System comparison of total windows examined based on the CMU test set.

Chapter 6

Discussion

Since SVM-based learning and classification for face detection can be computationally costly, a new classifier was introduced in chapter 2 which circumvents this problem by offloading a large share of the workload to a simpler classifier. The linear discriminant (FLD) classifier operates as a prefilter by passing only *difficult* patterns to the second-stage SVM classifier. The parameters T_a and T_b allow used-defined control of the degree to which the first-stage (and less accurate) classifier is used. Based on the CMU test set, Figure 5.10 illustrates how the two-stage classifier can be nearly as accurate as a single SVM-based classifier while significantly reducing the computational complexity of classification.

A second improvement to the system is made by incorporating a front-end selective attention mechanism into the system. The intent is to reduce the search area within an image by directing the classifier to significant target areas. While the work in this area is far from complete, the idea of performing face detection by integrating a selective attention mechanism inspired by the architecture of the primate visual

system is investigated here showing promising results. The results demonstrate how the model is effective at finding face targets in images with only a few faces using 50 targets for example. When images contain a larger number of faces, a much larger number of targets are needed, with degraded fidelity of each target. Since the targets are computed serially from the saliency map, a larger number of targets require a much longer delay in the face detection process. Using 50 targets or less required under a few seconds of processing time.

The results demonstrate satisfactory face detection rates based on the CMU test set. Tables 5.5 and 5.6 summarize the effectiveness of the proposed face detection system. In addition to displaying similar detection rates to other published methods, the main advantage is realized by the significant reduction in computational complexity.

The CMU test set represents a very challenging data set because the images tend to be very noisy with poor resolution in some cases since they are scanned from newspapers, and extracted from analog tv sequences. Therefore, the test set is a very good (and possibly extreme) representation of real-world face detection problems. Additionally, the results affirm the powerful capabilities of a support vector machine-based classifier by handling small variations in pose, translation and illumination of image-based features.

Despite the encouraging results from the system, there are a number of problems left unanswered concerning the selective attention system. The number of targets chosen somewhat arbitrarily as $10-50$ is limited mainly by the cycle time for computing the targets. The number of targets needed is largely unknown, with the only observation encountered being that images with a large number of faces in most cases always needed more than 50 targets.

Additionally, it would be interesting to see the effect of including a feedback mechanism into the selective attention system from a higher level supervisor (the classifier). Currently, the selective attention system provides targets independently, from a bottom-up fashion based solely on low-level features (color, intensity, orientation). Perhaps the quality of the targets could be improved by integrating a top-down mechanism from the classifier by providing real-time feedback based on the location of previous targets relative to classified faces.

Appendix A

Receiver Operating Characteristic Graphs

Receiver Operating Characteristic (ROC) graphs depict a cost/benefit analysis of decision making which is helpful in selecting an optimal model (i.e. classifier choice) independently from a cost (or "risk") context or the class distribution. They are especially useful in problems with unbalanced class distributions [15] such is the case of face/non-face classes in image-based face detection.

Given a classifier and a prediction instance, one of four outcomes are possible. If the instance is positive (ex: a face), and is classified as positive, the prediction is considered a true positive (TP). However, if it is classified as negative, it is counted as a false negative (FN). The same logic follows for the negative instance (non-face). If it is classified as negative, it is counted as a true negative (TN), but if it is classified as positive, it is counted as a false positive (FP). Clearly, in a decision making process, a high TP rate and low FP rate is desired.

Many classifiers (such as bayes, neural networks, SVMs) yield an instance probability or "score" representing the degree to which an instance is a member of a class. Such a ranking or scoring classifier (such as SVM and FLD classifiers) uses a threshold to produce a discrete (binary) classification output $\in \{-1, +1\}$. If the classifier output is above or below the threshold, the classifier produces a $+1$ or -1 (ex: face or non-face) respectively. Given a classifier arrangment (ex: SVM), each unique thresholded value defines a unique classifier instance and produces a different point in the ROC space. By varying the threshold from $-\infty$ to $+\infty$, a curve can be traced out in the ROC space.

ROC graphs are two-dimensional plots in which the TP rate is plotted against the FP rate. where

$$TP\ rate = \frac{\text{positives correctly classified}}{\text{total positives}} \tag{A.1}$$

and

$$FP\ rate = \frac{\text{negatives incorrectly classified}}{\text{total negatives}} \tag{A.2}$$

Algorithm 1 describes the conceptual idea of ROC curve calculation.

An ROC curve therefore illustrates the cost (false positives) and benefit (true positives) trade-offs. Or equivelently, it shows the ability of the classifier to rank the positive instances (ex: faces) relative to the negative instances (ex: non-faces). For example, some applications may require a high sensitivity to false positives, while others may only require a high positive detection rate. The ROC curve therefore provides a convenient way to describe this tradeoff for different models of the classifier by varying the threshold.

Figure A.1 illustrates a toy example of a typical ROC curve. There are several important points on any ROC curve worth noting. The lower left point $(0, 0)$ represents

Algorithm 1 Conceptual method for calculating ROC curve of classifier $f(x)$

Require: $\mathcal{D}$, the set of test patterns; $f(x)$ the classifiers estimated target for x; *min* and *max*, the minimum and maximum values returned by f; *increment*, the smallest difference between any two f values; P and N, the total number of positive and negative test patterns.

```
for threshold = min to max by increment do
  FP ← 0
  TP ← 0
  for x ∈ D do {process all test patterns}
    if f(x) ≥ threshold then
      if x is a positive example then {x is a "face"}
        TP ← TP + 1
      else {x is not a "face", therefore a false positive}
        FP ← FP + 1
      end if
    end if
  end for
  Add point (FP/N, TP/P) to ROC curve
end for
```

the classifier model that never commits a false positive error but also gains no true positives. The opposite model is represented by the point $(1, 1)$ where all patterns are classified as positive while committing false positive errors always. The diagonal line $y = x$ represents the model which randomly guesses a class. For example, the point $(0.6, 0.6)$ represents the classifier which guesses the positive class 60% of the time, and the negative class 40% of the time. Therefore, if the classifier guesses the positive class 60% of the time, it can expect to get 60% of the positives correct, but its false positive rate will be 60% as well.

The point $(0, 1)$ represents perfect classification (100% TP rate, 0% FP rate) and is the objective point for any good classifier. Therefore, points in the ROC space closer to $(0, 1)$ perform better overall than any other point. Classifiers corresponding to points towards $(0, 0)$ can be considered "conservative" classifiers since they make positive

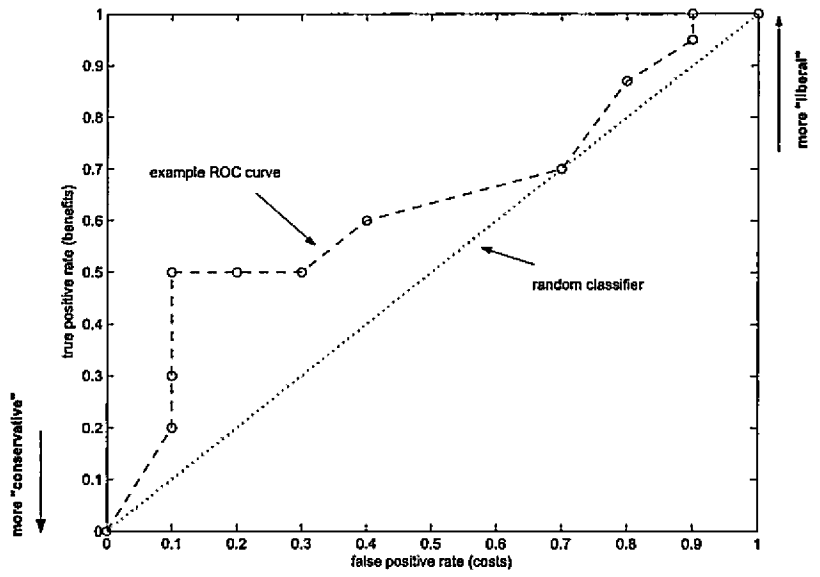


Figure A.1: example of ROC curve created by varying the threshold point for a given test set.

classifications only with strong evidence and therefore make few false positive errors at the expense of having a low false positive rate. Conversely, classifiers corresponding to points towards $(1, 1)$ are more "liberal" classifiers making positive classifications with weak evidence such that nearly all positives are classified correctly; but they have a corresponding high false positive rate [15].

Bibliography

[1] "Cbcl face database 1," *MIT Center For Biological and Computation Learning http://www.ai.mit.edu/projects/cbcl.*

[2] "Cif/qcif video sequences," *http://delboy.ucd.ie/research/downloads/test_sequences.*

[3] G. Antonini, V. Popovici, and J. Thiran, "Independent component analysis and support vector machine for face feature extraction," *AVBPA*, pp. 111–118, 2003.

[4] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996. [Online]. Available: citeseer.ist.psu.edu/breiman96bagging.html

[5] G. Burel and D. Carel, "Detection and localization of faces on digital images," *Pattern Recogn. Lett.*, vol. 15, no. 10, pp. 963–967, 1994.

[6] C. J. C. Burges, "Simplified support vector decision rules," in *International Conference on Machine Learning*, 1996, pp. 71–77.

[7] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[8] J. Cai and A. Goshtasby, "Detecting human faces in color images," *IVC*, vol. 18, no. 1, pp. 63–75, December 1999.

[9] C. Conde, A. Ruiz, and E. Cabello, "Pca vs low resolution images in face verification," in *CIAP03*, 2003, pp. 63–67.

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[11] E. H. Craw, I.G. and J. Lishman, "Automatic extraction of facial features," vol. 5, no. 2, 1987, pp. 183–187.

[12] N. Cristianini and J. Shawe-Taylor, *An Introduction To Support Vector Machines: And Other Kernel-Based Learning Methods.* Cambridge University Press, 2000.

[13] J. Crowley and F. Berard, "Multi-model tracking of faces for video communications," *IEEE Proc. of Int, Conf. on Computer Vision and Pattern Recognition*, Jun.

[14] R. Duda and P. Hart, *Pattern Classification and Scene Analysis.* New-York: John Wiley & Sons, 1973.

[15] T. Fawcett, "Roc graphs: Notes and practical considerations for data mining researchers," 2003. [Online]. Available: citeseer.ist.psu.edu/fawcett03roc.html

[16] A. Gammerman, *Computational Learning and Probabilistic Reasoning.* John Wiley & Sons, 1996.

[17] C. Garcia and M. Delakis, "A neural architecture for fast and robust face detection," *Proc. of the IEEE-IAPR International Conference on Pattern Recognition*, vol. 2, 2002.

[18] R. C. Gonzalez and R. E. Woods, *Digital Image Processing.* Addison-Wesley Longman Publishing Co., Inc., 2001.

[19] A. P.-T. S. C. Gupta, H. Agrawal and R. Chellappa, "An experimental evaluation of linear and kernel-based methods for face recognition," *Proceedings. Sixth IEEE Workshop on Applications of Computer Vision*, pp. 13–18, 2002.

[20] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, 1990.

[21] S. Haykin, *Neural Networks - A Comprehensive Foundation.* Prentice Hall, 1994.

[22] E. Hjelmas and B. Low, "Face detection: A survey," *CVIU*, vol. 83, no. 3, pp. 236–274, September 2001.

[23] R. Hsu, M. Abdel-Mottaleb, and A. Jain, "Face detection in color images," *PAMI*, vol. 24, no. 5, pp. 696–706, May 2002.

[24] W. Huitt, "The information processing approach to cognition," *Educational Psycology Interactive*, 2003.

[25] L. Itti and C. Koch, "Computational modelling of visual attention," *Nature Reviews Neuroscience*, vol. 2, pp. 1–10, 2001.

[26] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, Nov 1998. [Online]. Available: http://klab.caltech.edu/ itti/attention/publications/98_PAMI/

[27] H.-M. Je, D. Kim, and S. Yang Bang, "Human face detection in digital video using svmensemble," *Neural Process. Lett.*, vol. 17, no. 3, pp. 239–252, 2003.

[28] I. T. Jolliffe, *Principal Component Analysis.* Springer, 1986.

[29] C. Koch and S. Ullman, "Shifts in selective visual attention: Towards the underlying neural circuitry," *Human Neurobiology*, vol. 4, pp. 219–227, 1985.

[30] K. Lam and H. Yan, "Facial feature location and extraction for computerised human face recognition," *International Symposium on information Theory and Its Applications*, Nov.

[31] A. Leventhal, "The neural basis of visual functions: Vision and visual dysfunction," vol. 4, 1991.

[32] Y. Li, S. Gong, J. Sherrah, and H. Liddell, "Support vector machine based multi-view face detection and recognition," *IVC*, vol. 22, no. 5, pp. 413–427, May 2004.

[33] J. Maydt and R. Lienhart, "A fast method for training support vector machines with a very large set of linear features," in *Proceedings of IEEE ICME 2002*, Lousanne, Switzerland, 2002.

[34] G. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition.* John Wiley & Sons Inc., 1992.

[35] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding: CVIU*, vol. 81, no. 3, pp. 231–268, 2001. [Online]. Available: citeseer.ist.psu.edu/moeslund01survey.html

[36] S. Nayar, S. Nene, and H. Murase, "Subspace methods for robot vision," *RA*, vol. 12, no. 5, pp. 750–758, October 1996.

[37] J. Ng and S. Gong, "Composite support vector machines for detection of faces across views and pose estimation," *IVC*, vol. 20, no. 5-6, pp. 359–368, April 2002.

[38] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: An application to face detection," in *CVPR97*, 1997, pp. 130–136.

[39] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," pp. 185–208, 1999.

[40] V. Popovici and J. Thiran, "Face detection using an svm trained in eigenfaces space," in *AVBPA03*, 2003, pp. 190–198.

[41] B.-B. E. Popovici V, Thiran J, "The banca database and evaluation protocol," in *4th International Conference on Audio- and Video-Based Biometric Person Authentication, Guildford, UK*, ser. Lecture Notes in Computer Science, vol. 2688. Berlin: Springer-Verlag, June 2003, pp. 625–638.

[42] M. L. R. Xiao and H. Zhang, "Robust multipose face detection in images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, 2004.

[43] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *PAMI*, vol. 20, no. 1, pp. 23–38, January 1998.

[44] K. K. S. Theodoridis, *Pattern Recognition, 2nd Edition.* Academic Press, 2003.

[45] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2001.

[46] C. Siagian and L. Itti, "Biologically-inspired face detection: Non-brute-force-search approach," in *First IEEE International Workshop on Face Processing in Video*, June 2004, cv — bu — sc.

[47] K.-K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 39–51, 1998.

[48] Y. Y. T. Yokoyama and M. Yachida, "Facial contour extraction model," *IEEE Proc. of 3rd Int. Conf. on Automatic Face and Gesture Recognition*, 1998.

[49] M. Turk, "A random walk through eigenspace," *IEICE Trans. Inf. & Syst.*, vol. E84-D, no. 12, pp. 1586–1595, December 2001.

[50] M. Turk and A. Pentland, "Eigenfaces for recognition," *CogNeuro*, vol. 3, no. 1, pp. 71–96, 1991.

[51] V. Vapnik, *The nature of statistical learning theory.* New York: Springer, 1995.

[52] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[53] M. Yang, D. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *PAMI*, vol. 24, no. 1, pp. 34–58, January 2002.