

# Hybrid Systems for Face Recognition

---



Master of Science Graduate Thesis  
Written by Atle Nes ([atlen@idi.ntnu.no](mailto:atlen@idi.ntnu.no))  
Supervised by Kjetil Bø ([ketilb@idi.ntnu.no](mailto:ketilb@idi.ntnu.no))

Submitted 16<sup>th</sup> of June 2003

---

Norwegian University of Science and Technology  
Faculty of Information Technology,  
Mathematics and Electrical Engineering  
Department of Computer and Information Science  
Division of Intelligent Systems / Image Processing

## Summary

Face recognition is an example of advanced object recognition. The process is influenced by several factors such as shape, reflectance, pose, occlusion and illumination which make it even more difficult. Today there exist many well known techniques to try to recognize a face. We present to the reader an investigation into individual strengths and weaknesses of the most common techniques including feature based methods, PCA based eigenfaces, LDA based fisherfaces, ICA, Gabor wavelet based methods, neural networks and hidden Markov models.

Hybrid systems try to combine the strengths and suppress the weaknesses of the different techniques either in a parallel or serial manner. Different combinations of the techniques described above are evaluated. A hybrid combination using Gabor wavelets for coarse sorting and PCA for fine sorting is described and implemented. The implementation has been done exclusively using Matlab and its Image Processing Toolbox.

Finally we do some testing on the FERET face database to see how good our implementation is. Both frontal head images and rotated head images are investigated.

## Table of Contents

<b>1</b>	<b>PREFACE .....</b>	<b>5</b>
<b>2</b>	<b>VOCABULARY.....</b>	<b>6</b>
<b>3</b>	<b>INTRODUCTION .....</b>	<b>12</b>
3.1	PROJECT BACKGROUND .....	12
3.2	ASSIGNMENT DESCRIPTION.....	12
<b>4</b>	<b>EVALUATING THE APPROACHES.....</b>	<b>13</b>
4.1	GEOMETRY AND TEMPLATES .....	13
4.1.1	<i>Introduction.....</i>	13
4.1.2	<i>Geometric Methods.....</i>	13
4.1.3	<i>Template Matching.....</i>	15
4.1.4	<i>Dynamic Deformable Templates .....</i>	15
4.1.5	<i>Evaluation .....</i>	16
4.2	PRINCIPAL COMPONENT ANALYSIS (PCA).....	17
4.2.1	<i>Eigenfaces .....</i>	17
4.2.2	<i>Eigenfeatures.....</i>	19
4.2.3	<i>PCA in the Fourier Domain .....</i>	19
4.2.4	<i>Eigen Light-Fields.....</i>	20
4.2.5	<i>Evaluation .....</i>	21
4.3	LINEAR DISCRIMINANT ANALYSIS (LDA).....	23
4.3.1	<i>Fisherfaces .....</i>	23
4.3.2	<i>LDA in the Fourier Domain .....</i>	24
4.3.3	<i>Fisher Light-Fields.....</i>	24
4.3.4	<i>Evaluation .....</i>	25
4.4	INDEPENDENT COMPONENT ANALYSIS (ICA) .....	26
4.4.1	<i>Introduction.....</i>	26
4.4.2	<i>Tensorfaces.....</i>	27
4.4.3	<i>Evaluation .....</i>	28
4.5	WAVELETS.....	29
4.5.1	<i>Introduction.....</i>	29
4.5.2	<i>Gabor Wavelets .....</i>	29
4.5.3	<i>Elastic Bunch Graph Matching .....</i>	31
4.5.4	<i>Gabor Fisher Classifier (GFC) .....</i>	32
4.5.5	<i>Evaluation .....</i>	32
4.6	HIDDEN MARKOV MODELS (HMM).....	34
4.6.1	<i>Introduction.....</i>	34
4.6.2	<i>One-Dimensional HMM .....</i>	34
4.6.3	<i>Pseudo and Embedded Two-Dimensional HMM.....</i>	35
4.6.4	<i>Evaluation .....</i>	38
4.7	NEURAL NETWORKS .....	40
4.7.1	<i>Introduction.....</i>	40
4.7.2	<i>Multi-Layered Feed-Forward Networks.....</i>	40
4.7.3	<i>Radial Basis Function (RBF) Networks.....</i>	42
4.7.4	<i>Dynamic Link Architecture (DLA).....</i>	42
4.7.5	<i>Evaluation .....</i>	43
<b>5</b>	<b>HYBRID SYSTEMS.....</b>	<b>44</b>
5.1	INTRODUCTION .....	44
5.2	HYBRID PARALLEL APPROACHES.....	44
5.2.1	<i>Combining Frontal and Profile Templates.....</i>	44
5.2.2	<i>Combining LDA and PCA .....</i>	45
5.2.3	<i>Combining HMM, PCA and Profile Templates .....</i>	46
5.2.4	<i>Combining with Other Biometrics .....</i>	47
5.3	HYBRID SERIAL APPROACH .....	47
5.3.1	<i>ERBF and Decision Trees .....</i>	47
5.3.2	<i>Neural Network and HMM .....</i>	48

<b>6</b>	<b>IMPLEMENTATION .....</b>	<b>49</b>
6.1	CHOICE OF HYBRID COMBINATION .....	49
6.1.1	<i>Normalization Module</i> .....	50
6.1.2	<i>Coarse Primary Screening Module</i> .....	50
6.1.3	<i>Fine Secondary Screening Module</i> .....	52
6.2	SOFTWARE ENGINEERING TOOLS .....	53
6.2.1	<i>Matlab 6.5</i> .....	53
6.2.2	<i>Image Processing Toolbox 4</i> .....	54
6.3	SYSTEM MODULES .....	55
6.3.1	<i>Face Normalization (facenorm.m)</i> .....	55
6.3.2	<i>Batch Face Normalization (facenormall.m)</i> .....	58
6.3.3	<i>Top of Face Normalization (facenormtop.m)</i> .....	58
6.3.4	<i>Batch Top of Face Normalization (facenormtopall.m)</i> .....	58
6.3.5	<i>View-Based Face Normalization (facenormrot.m)</i> .....	59
6.3.6	<i>Gabor Training (gabortrain.m)</i> .....	61
6.3.7	<i>Gabor Recognition (gaborreq.m)</i> .....	63
6.3.8	<i>PCA Training (eigenrain.m)</i> .....	64
6.3.9	<i>PCA Recognition (eigenreq.m)</i> .....	65
6.3.10	<i>Test Gabor Wavelet Recognizer (testgabor.m)</i> .....	66
6.3.11	<i>Test Hybrid Combination Recognizer (testhybrid.m)</i> .....	66
<b>7</b>	<b>RESULTS .....</b>	<b>67</b>
7.1	NORMALIZING THE WHOLE FACE OR JUST THE TOP? .....	67
7.2	HOW MANY GABOR JET SAMPLES IN THE RECTANGULAR GRID? .....	68
7.3	TUNING THE GABOR MODULE .....	69
7.4	L2NORM, MAHALANOBIS OR A COMBINED DISTANCE MEASURE? .....	70
7.5	HEAD ROTATED FACE IMAGES .....	71
7.6	ELAPSED TIME .....	71
<b>8</b>	<b>CONCLUSION .....</b>	<b>72</b>
<b>9</b>	<b>REFERENCES .....</b>	<b>73</b>
9.1	LITERATURE .....	73
9.2	HYPERLINKS .....	81
<b>10</b>	<b>INDEXES .....</b>	<b>82</b>
10.1	FIGURES .....	82
<b>11</b>	<b>APPENDIX .....</b>	<b>83</b>
11.1	FACENORM.M .....	83
11.2	FACENORMROT.M .....	84
11.3	GABORTRAIN.M .....	86
11.4	GABORREC.M .....	88
11.5	EIGENTRAIN.M .....	89
11.6	EIGENREC.M .....	91
11.7	TESTHYBRID.M .....	92

## 1 Preface

This is the result of my Master of Science graduate thesis surrounding image processing and intelligent systems at the Department of Computer and Information Science, Norwegian University of Science and Technology. The assignment was posted and supervised by Kjetil Bø.

This thesis has been completed during 20 weeks in spring 2003. It has dominated this semester completely, and I feel the work has given me much valuable knowledge and experience. I would like to use this opportunity to thank my supervisor Kjetil Bø for good response and support.

## 2 Vocabulary

This chapter can be used for inquiry if a phrase or a word in the following report needs any further explanation.

### **Backpropagation Algorithm**

Is a method for training a supervised neural network. Backpropagation is used to imply a backward pass of error to each internal node within the network, which is then used to calculate weight gradients for that node.

### **Baum-Welch Algorithm**

An algorithm to find hidden markov model parameters  $A$ ,  $B$ , and  $\Pi$  with the maximum likelihood of generating the given symbol sequence in the observation vector.

### **Biometrics**

Automatic identification or verification of human beings using biological characteristics belonging to the individual.

### **Chess Board Distance**

$$d_{\text{chessboard}} = \max(|x_2 - x_1|, |y_2 - y_1|)$$

### **City Block Distance**

$$d_{\text{cityblock}} = |x_2 - x_1| + |y_2 - y_1|$$

### **Correlation**

Is a measure of the association strength of the relationship between two variables.

### **DCT - Discrete Cosine Transform**

Helps separate the image into parts or spectral sub-bands of differing importance with respect to the image's visual quality. The DCT is similar to the discrete Fourier transform. It transforms a signal or image from the spatial domain to the frequency domain.

### **DLA - Dynamic Link Architecture**

Is a neural network where the image and all the models are represented by layers of neurons labelled by jets as local features.

### **Eigenfaces**

Are PCA based eigenvectors of the face.

### **Eigenfeatures**

Are eigenvectors of features like eyes, nose or mouth.

### **Eigenspace**

Subspace spanned by the eigenvectors of the covariance matrix.

### **Energy Function**

The energy function is a function defined over conformation space and associates each possible conformation with its energy.

**Euclidean Distance**

$$d_{\text{euclidean}} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

**FaceIt**

One of the leading commercial face recognition software packages made by Visionics. The recognition software uses a feature based approach with several face distance measures and template matching.

**FBG - Face Bunch Graph**

Faces may have a beard or glasses, may have different expressions, or may be of different age, sex, or race. The Face Bunch Graph has a stack-like structure and combines graphs of individual sample faces. It is crucial that the individual graphs all have the same structure and that the nodes refer to the same fiducial points.

**Feature Space**

A space formed by feature vectors.

**Feature Vector**

Is a vector containing information about a specific observed feature.

**Feed Forward Network**

Neural network consisting of input layers, one or more hidden layers and one output layer with nodes. These nodes are connected from input nodes via hidden layers to output nodes. No feedback connections are allowed.

**Fiducial Point**

Are landmark points in the image used for recognition. This can be eyes, nose, mouth, ear lobes etc.

**FLD - Fisher Linear Discriminant**

See LDA - Linear Discriminant Analysis.

**Fisherfaces**

Are LDA based eigenvectors of the face.

**Fourier Transform**

Decomposes or separates a waveform or function into sinusoids of different frequency which sum to the original waveform. It identifies or distinguishes the different frequency sinusoids and their respective amplitudes

**Gabor Jet**

Is a set of 2D gabor responses obtained when convoluting gabor wavelets of different rotations and scales.

**Gabor Wavelet**

2D gabor wavelets are biological motivated convolution kernels in the shape of plane waves restricted by a Gaussian envelope function.

**HMM - Hidden Markov Model**

A variant of a finite state machine having a set of states  $Q$ , an output alphabet  $O$ , transition probabilities  $A$ , output probabilities  $B$ , and initial state probabilities  $\Pi$ . The current state is not observable. Instead, each state produces an output with a certain probability  $B$ . Usually the states  $Q$ , and outputs  $O$ , are understood, so an HMM is said to be a triple  $(A, B, \Pi)$ .

**Hybrid**

Hybrid systems try to combine the strengths and suppress the weaknesses of the different techniques either in a parallel or serial manner.

**ICA - Independent Component Analysis**

Is a technique for extracting statistically independent variables from a mixture of them. The technique is quite new and has originated from the world of signal processing.

**JPEG - Joint Photographic Experts Group**

Image format originally designed to transfer graphic data and images via digital telecommunication networking and was generally used to hold and transfer full color photorealistic images. JPEG compresses photos though with quality loss.

<http://www.jpeg.org/>

**KLT - Karhunen-Loeve Transform**

See PCA – Principal Component Analysis.

**Labelled Graph**

A labelled graph is built by assigning labels successively to vertices or edges after the unlabelled graph has been constructed.

**LDA - Linear Discriminant Analysis**

Finds the line that best separates the points. In terms of face recognition this means grouping images of the same class and separate images of different classes.

**Light-Field**

The plenoptic function or light-field specifies the radiance of light from an object seen from every position outside the object in every direction. It is typically assumed to be a 5D function, consisting of position (3D) and orientation (2D). It is also sometimes modeled as a function of wavelength, polarization and time.

**Matlab**

Matlab is a simulation environment for doing numerical computations with matrices and vectors.

**MLP – Multi-Layer Perceptron**

Is a neural network good for classification purposes.

**Module**

A module is a part of a system that performs a task.



**Mother function**

Is a basic wavelet function where other functions can be obtained by translation and dilation of this mother function.

**MPEG - Moving Pictures Experts Group**

Is the name of family of standards used for coding audio-visual information (movies, video, music) in a digital compressed format.

**MS - Microsoft**

The world largest producer of PC software. Founded in 1975 by Paul Allen and Bill Gates, two college students who together wrote the first Basic interpreter for Intel's 8080-microprocessor. Microsoft is most famous for its operating systems MS-DOS and MS Windows. (<http://www.microsoft.com/>)

**Neural Network**

At the core of neural computation are the concepts of distributed, adaptive, and nonlinear computing. Neural networks perform computation in a very different way than conventional computers, where a single central processing unit sequentially dictates every piece of the action. Artificial neural networks have provided solutions to problems normally requiring human observation and thought processes.

**PCA - Principal Component Analysis**

Means rotating the data so that its primary axes lie along the axes of the coordinate space and move it so that its center of mass lies on the origin.

**PGM - Portable Grey Map**

The PGM format is a lowest common denominator greyscale image file format. It is designed to be extremely easy to learn and write programs for.

**Plenoptic Function**

See Light-Field.

**RBF - Radial Basis Function**

Are neural networks with fast learning speed.

**Scatter Matrix Analysis**

Used in LDA. Analysis of scatter between different classes and scatter of samples within the class.

**Subspace**

An image can be seen as a vector of pixels where the value of each entry in the vector represents the grey value of the image pixel intensity. This means that an image with size 8x8 pixels can be viewed as a vector of size 64. The image is then represented in an N-dimensional space where N is the length of this vector. This N-dimensional vector space is called original space and is only one of many subspaces which can be used to represent the image.

**SVD - Singular Value Decomposition**

A widely used technique to decompose a matrix into several component matrices, exposing many of the useful and interesting properties of the original matrix. SVD allows one to diagnose the problems in a given matrix and provides numerical answers as well.

**Template Matching**

A distance function (typically a simple Euclidean distance) is applied to measure the similarity of the template and the image at the location. The algorithm then picks the location with smallest distance as the location of the template image in the target image.

**Tensorfaces**

Is a modelling technique also known as N-node singular value decomposition (SVD). Tensor decompositions can be used in conjunction with higher order statistics employed in ICA.

**Test image**

Test images are a set of images unknown to the computer. These are images that we want to identify. In the recognition stage test images are compared to the known training images stored in the computer database.

**Threshold**

Operation used to correct abberating pixel values in an image.

**Training image**

Training images are a set of images known to the computer. These are images that are already identified. In the recognition stage training images are stored in a database and used for comparison to try to identify the test images.

**Viterbi Segmentation**

An algorithm to compute the most likely state sequence in a hidden markov model given a sequence of observed outputs.

**Wavelet**

The fundamental idea behind wavelets is to analyze according to scale. Wavelets are functions that satisfy certain mathematical requirements and are used in representing data or other functions.

**Windows**

The most widespread operating system for PCs. Developed by Microsoft Corporation. Windows has a graphical user interface, and comes in several different versions. The most used today are Windows 95/98/Me and Windows NT4/2000/Xp.

(<http://www.microsoft.com/windows/>)

### 3 Introduction

This chapter will give some background information about biometrics. The project assignment text, intention and limitations will also be defined.

#### 3.1 Project Background

Face recognition is an example of advanced object recognition. The process is influenced by several factors such as shape, reflectance, pose, occlusion and illumination. A human face is an extremely complex object with features that can vary over time, sometimes very rapidly. It is covered with skin, a non-uniformly textured material that is difficult to model. Skin can change colour quickly when one is embarrassed or becomes warm or cold and the reflectance properties of the skin change as the perspiration level changes. The face is a highly deformable object, and facial expressions come in a wide variety of possible configurations. Time-varying changes include growth and removal of facial hair, wrinkles and sagging of the skin caused by aging and change in skin colour because of exposure to sunlight. Artifact-related changes include cuts, scrapes and bandages from injuries and fashion-related issues like makeup, jewelry and piercings. It should be quite clear that the human face is much more difficult to model and recognize than most industrial parts. This hard challenge is one of the reasons why computer vision research community has been devoted to face recognition for quite some time.

Access control by face recognition has the following advantages in comparison with other biometric systems. There are no requirements for expensive or specialized equipment. A system can be built using a simple video camera and a personal computer. Another advantage is that it is a passive system. There is no need for individuals to touch something by fingers or palm, no need to say any word or lean eye to a detector. Any person can just walk or stay before the camera and the system performs recognition. It is especially useful in everyday usage. Also it has advantages in different extreme and non-standard situations, for example when catching criminals.

#### 3.2 Assignment Description

This report builds on the solid theoretical platform made in the course SIF8092 Image Processing Specialization Project from the previous semester [1]. The implementation to test the hybrid system has been done using Matlab. This will be described thoroughly later.

##### **The original assignment text:**

*Hybrid Systems for Face Recognition*

*Today there exist many well known techniques to try to recognize a face. Experiments done with implementations of different methods have shown that they have individual strengths and weaknesses. Hybrid systems try to combine the strengths and suppress the weaknesses of the different techniques either in a parallel or serial manner. The assignment is to evaluate the different techniques and consider different combinations of these.*

## 4 Evaluating the Approaches

This chapter will give in-depth information about and evaluation of the most important computer vision face recognition methods that exist today. The focus here is on trying to find advantages and disadvantages with the different approaches. First we start by looking at some of the earliest approaches using simple feature based methods. Then we take a look at some more sophisticated statistical and holistic methods like PCA (eigenfaces), LDA (fisherfaces) and ICA. Other methods discussed are Gabor wavelets, hidden Markov models and neural networks.

### 4.1 Geometry and Templates

#### 4.1.1 Introduction

The earliest approaches to face recognition were focused on detecting individual features such as eyes, ears, head outline and mouth, and measuring different properties such as size, distance and angles between features. This data was used to build models of faces and made it possible to distinguish between different identities. This kind of system was proposed by Kanade in 1973 [5] and was one of the first approaches to automated face recognition. Later work by Yuille, Cohen and Hallinan in 1989 describes a method for feature extraction using deformable templates [6].

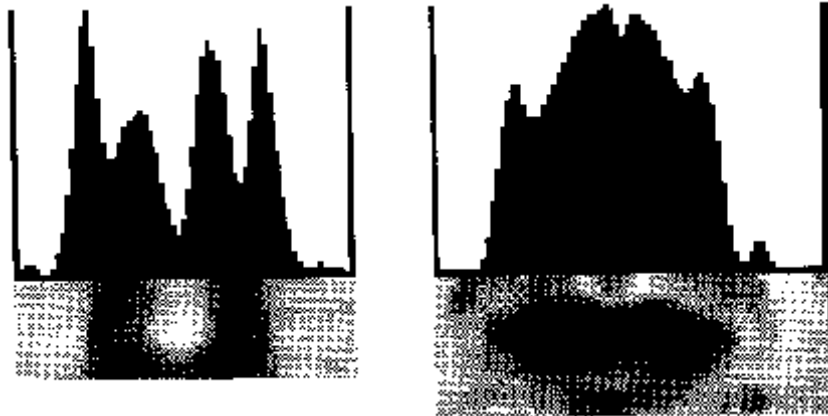
#### 4.1.2 Geometric Methods

Brunelli and Poggio developed two simple algorithms for face recognition [7]. The first one is based on the computation of a set of geometrical features, such as nose width and length, mouth position and chin shape. One motivation for using geometric methods is that in an image with sufficiently low resolution it is impossible to distinguish the fine details of a face, but often possible for a human to recognize the person. The remaining information in the low resolution image is almost pure geometrical and implies that these properties of face features are sufficient enough for face recognition. The configuration of the features can be described by a vector of numerical data representing the position and size of the main facial features, eyes and eyebrows, nose and mouth. This information can be supplemented by the shape of the face outline.

One of the most critical issues in using a vector of geometrical features is proper normalization. The extracted features have to be independent of position, scale and rotation of the face in the image plane. Translation dependency can be eliminated once the origin of coordinates is set to a point that can be detected with good accuracy in each image. Rotation invariance can be achieved by horizontally aligning the eye-to-eye axis and scale invariance by using the distance between the two eyes. Locating the eyes is usually performed using templates for each of the eyes. Templates are described in the next section.

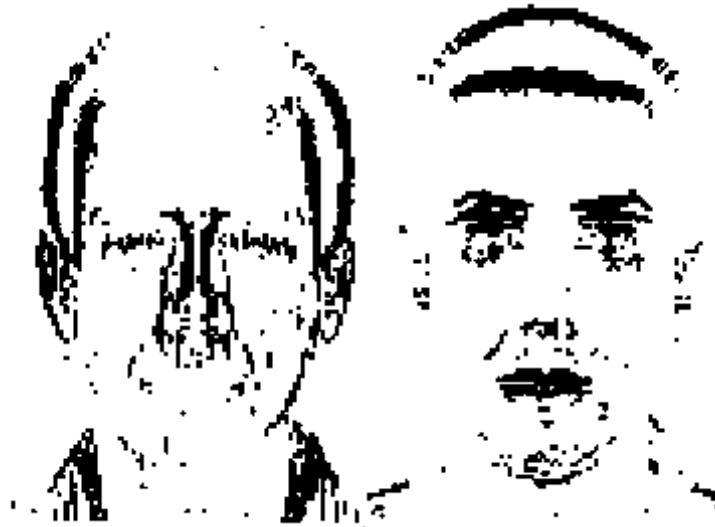
Because almost every face has two eyes, one nose and one mouth with very similar layout face classification can be difficult, while feature extraction is easier. A very useful technique for extraction of facial features is vertical and horizontal integral projection (Figure 1). Projections can be extremely effective in determining the

position of features, provided that the window on which they act is suitably located to avoid misleading interferences.



**Figure 1: Typical edge projections data**

The images are being preprocessed with two gradient operators to obtain edge images in horizontal and vertical direction, and then thresholded to make binary images (Figure 2). Horizontal gradients are useful to detect left and right boundaries of face and nose, whereas vertical gradients are useful to detect head top, eyes, nose base and mouth. The features of the face are then located by running integral projections in the areas where the different features are suspected to be, based on the location of the eyes and a priori knowledge of the average human physiology.



**Figure 2: Edge dominance maps (a) Horizontal (left) (b) Vertical (right)**

For example the nose can be located by looking for peaks of the horizontal projection of the vertical edge map. Mouth can be located by looking for a valley in the horizontal projection of the horizontal edge map due to the dark line between the lips. Eyebrows can be located using a similar strategy, and for detection of the face outline dynamic programming can be used to follow the outline on a gradient intensity map of an elliptical projection of the face image. Brunelli and Poggio used a total of 35

automatically extracted geometrical features for each face stored in a large 35-dimensional numerical vector.

### 4.1.3 Template Matching

The other algorithm proposed by Brunelli and Poggio is based on template matching. In the simplest version of template matching the image, represented by an array of intensity values, is compared using a suitable metric (typically Euclidean distance) to a single template representing the whole face. More sophisticated methods can use several templates per face to take into account the recognition from different viewpoints.

First the image is normalized using the same technique described in the previous section. Each person is stored in the database associated with template masks representing digital images of eyes, nose, mouth etc. Recognition of an unclassified image is done by comparing parts of it with all the templates stored in the database returning a matching score for each individual. The unknown individual is then classified as the one giving the highest cumulative comparison score.

### 4.1.4 Dynamic Deformable Templates

Yuille, Cohen and Hallinan proposed a method used deformable templates to detect and recognize faces [6]. The feature of interest, an eye for example, is described by a parameterized template. Templates give a priori information about the expected shapes and features of faces. They are flexible enough to change their size and other parameters to describe the features. Variations of the parameters should allow the template to fit any normal instance of the feature despite image variations in scale, tilt and rotation of head and lighting conditions.

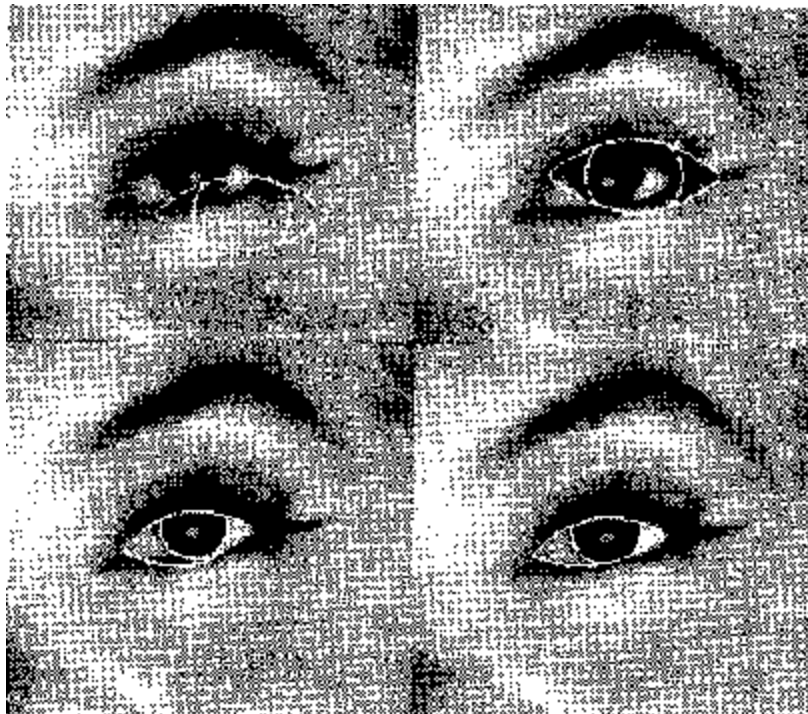


Figure 3: Dynamic sequence for eye detection

An energy function is defined which links edges, peaks and valleys in the image intensity to corresponding properties of the template. The template then interacts dynamically with the image, by altering its parameter values to minimize the energy function, thereby deforming itself to find the best fit (Figure 3). The final parameter values can be used as descriptors for the feature.

#### 4.1.5 Evaluation

The use of feature vectors seems very unstable and limited because the variation of the data from different pictures of the same face was in the same order of magnitude as the variation between different faces. The method is sensitive to inaccurate detection of features and to all sorts of disturbance such as facial expressions or varying pose.

Template-based approaches outperform geometrical methods. Templates seem to offer satisfactory results for recognition from frontal views. A more difficult problem is how to deal with non-frontal views. It should be possible to use almost the same scheme for different viewpoints at the expense of considerably greater computational complexity. Or maybe it is possible to extrapolate or guess correctly other views of the face. Humans are certainly able to recognize faces turned 20-30 degrees from the front from just one frontal view.

The recognition rate achieved with a single template (eyes, nose or mouth) is remarkable and consistent with the human ability of recognizing familiar people from a single facial characteristic. Using a eyes, nose or mouth template is most discriminating and using the whole face gives least discrimination. Integration of more features in a recognition system has a beneficial effect on robust classification. If more templates are used in parallel the score from the most similar feature can be used, scores can be added together or each feature template can be assigned a different weight.

<b>Advantages</b>	<b>Drawbacks</b>
<ul style="list-style-type: none"> <li>+ Robust against scaling, orientation and translation when face is correctly normalized</li> <li>+ Can handle high resolution images efficiently</li> <li>+ Saves neighbourhood relationships between pixels</li> <li>+ Can handle very low resolution images</li> <li>+ Geometric relations are stable under varying illumination conditions</li> <li>+ Good recognition performance</li> </ul>	<ul style="list-style-type: none"> <li>- Sensitive to faulty normalization</li> <li>- Sensitive to noise and occlusion</li> <li>- Templates are sensitive to illumination</li> <li>- Sensitive to perspective, viewing angle and head rotation (can be improved using more templates)</li> <li>- Sensitive to facial expressions, glasses, facial hair, makeup etc.</li> <li>- Slow training and recognition/High computational complexity</li> </ul>



## 4.2 Principal Component Analysis (PCA)

### 4.2.1 Eigenfaces

PCA also known as Karhunen-Loeve (KL) transformation or eigenspace projection, a frequently used statistical technique for optimal lossy compression of data under least square sense, provides an orthogonal basis vector-space to represent original data. The first introduction of a low-dimensional characterization of faces was developed at Brown University by Kirby and Sirovich in 1987 [8][9]. This was later extended to eigenspace projection for face recognition by Pentland, Turk, Moghaddam and Starner at the Vision and Modeling Group of MIT in 1991 [10][11][12][13]. More recently Nayar, Nene and Murase used eigenspace projection to identify objects using a turntable to view objects at different angles [14][15]. Finlayson, Dueck, Funt and Drew extended greyscale eigenfaces to colour images [16].

A two-dimensional image  $I(x,y)$  with  $N$  pixels may be viewed as a point or vector in a  $N$ -dimensional space, called image space. With this representation the image becomes a very high dimensional feature where modified traditional operations performed on feature vectors now can be used to manipulate the images directly. Increased resolution means increased dimensionality. Fortunately many key calculations scale with the number of sample images rather than the dimensionality of the image space, allowing efficiency even with relatively high resolution images.

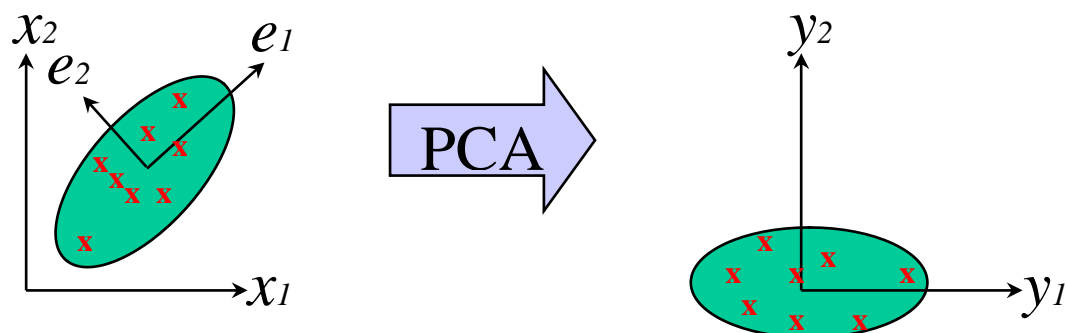
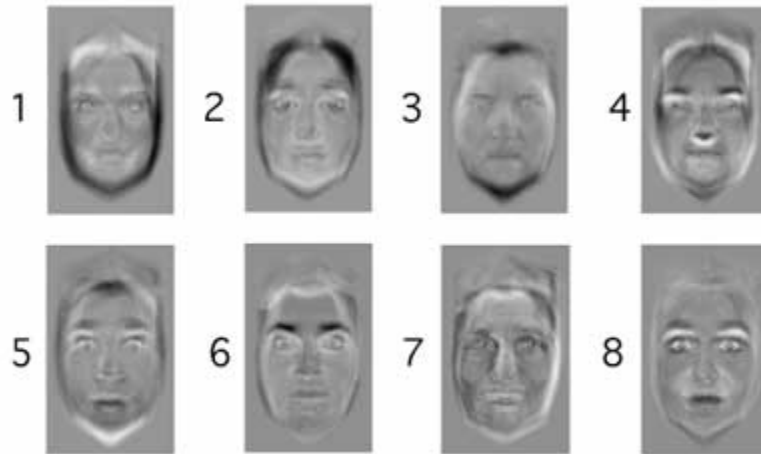


Figure 4: Principal Component Analysis (PCA)

The eigenspace is a subspace of the image space spanned up by a set of eigenvectors of the covariance matrix of the trained images. These eigenvectors are also called eigenfaces because of their face-like appearance (Figure 5). The covariance matrix is constructed by performing PCA which means rotating the dataset so that its primary axes, the eigenvectors with the highest modes of variation, lie along the axes of the coordinate space and move it so that its centre of mass corresponds with the origin (Figure 4). Eigenvectors with the highest associated eigenvalues represent the highest modes of variation in the dataset of images, and the eigenvectors with the lowest eigenvalues represent the lowest modes of variation. The dimensionality reduction to subspace can be performed in either a lossy or lossless manner. When applied in a lossy manner, eigenvectors are truncated from the front or back of the covariance matrix. It is assumed that these vectors correspond to not useful information such as lighting variations when dropped from the front or noise when dropped from the back. If none of the basis vectors are dropped it is called a lossless transformation and

it should be possible to get perfect reconstruction for the training data based on the compressed feature vectors. The projection of an image into eigenspace will transform the image into a representation of a lower dimension which aims to hold the most important features of the face and make the comparison of images easier.



**Figure 5: Eigenfaces have a face-like appearance**

There are two main approaches of recognizing faces by using eigenfaces [17]. In the appearance model (Figure 6a) each face in the database is represented as a linear combination of eigenfaces. The recognition process is done by projecting a test image to be identified into the same eigenspace. The resulting vector will be a point in eigenspace and comparison with the training images is normally done by using a distance measure between the points in eigenspace. There are different methods for calculating this distance, including simple measures like cityblock distance (L1 norm), chessboard distance, covariance, correlation, Mahalanobis distance and in this case most commonly Euclidean distance (L2 norm). The image with the shortest distance to the test image can be regarded as a match if the distance is below a preset threshold. Otherwise the image is regarded as unknown, and can possibly be added to a future training set.

The other recognition scheme is called the discriminative model (Figure 6b). Two datasets are obtained by computing intrapersonal differences (matching two different images of the same individual in the dataset) and extrapersonal differences (matching different individuals in the dataset). Two datasets of eigenfaces are generated by performing PCA on each class and a similarity score between two images is derived by calculating a Bayesian probability measure.

Although the recognition performance of the appearance model is lower than the discriminative model, the substantial reduction in computational complexity makes this recognition scheme very attractive.

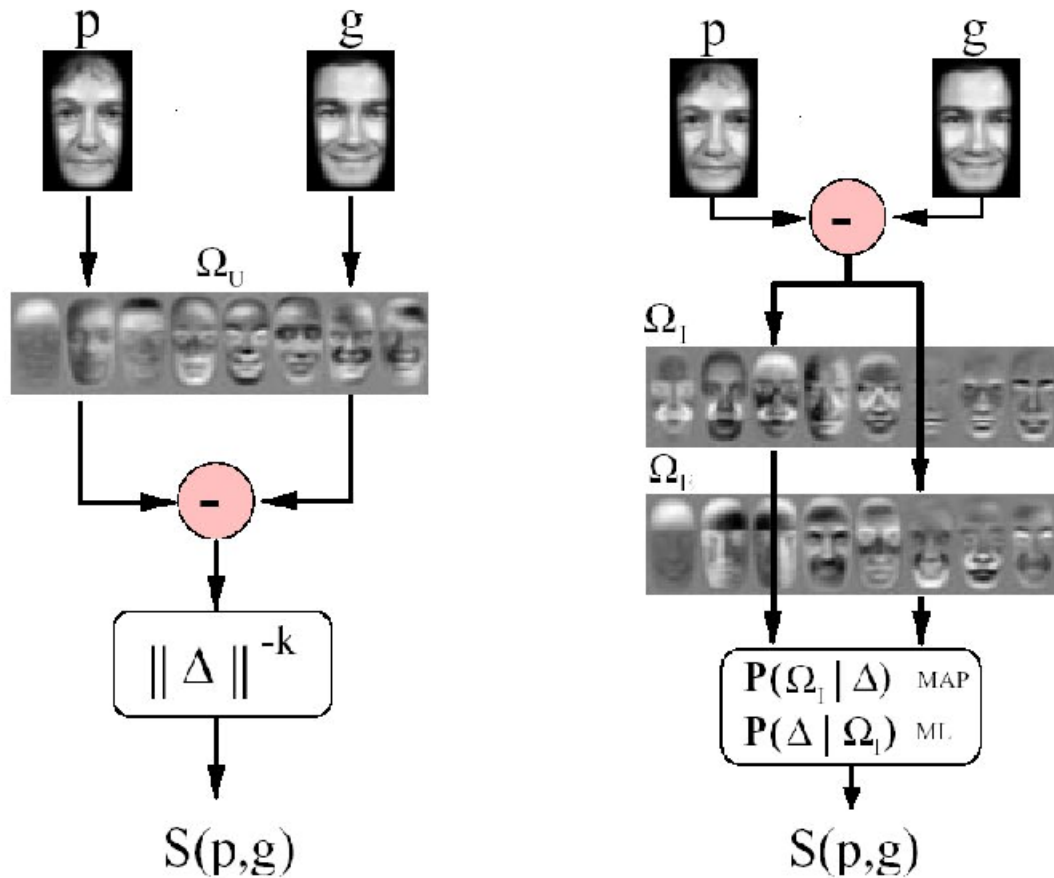


Figure 6: Eigenspace recognition (a) appearance model (left) (b) discriminative model (right)

#### 4.2.2 Eigenfeatures

Modular eigenspace, also called eigenfeatures, is an extended technique that uses not only the face as a whole like the standard eigenfaces method, but smaller features like eyes, nose, mouth etc. The computation is the same as with eigenfaces method, but will end up with eigeneyes, eigen noses and eigenmouths instead [12]. Eigenfeatures can be used as a preprocessing step for detecting the eye coordinates needed for normalization or face recognition directly [18]. Campos, Feris and Cesar even report better results using eigenfeatures from eyes, nose and mouth than using the whole face alone [19]. It should be noted that their training set consisted of only 15 people. Inclusion of nose and mouth regions sometimes reduces the performance because face expressions can make strong distortions in these regions.

#### 4.2.3 PCA in the Fourier Domain

Akamatsu, Sasaki and Suenaga have shown the effectiveness of PCA of the Fourier spectrum [30]. The eigenface method is applied to the magnitude of the Fourier spectrum of the normalized images. The method showed better performance than the classical eigenface algorithm when it comes to variations in head orientation and shifting. However the computational complexity of this method is significantly greater than the eigenface method due to the computation of the Fourier spectrum.

#### 4.2.4 Eigen Light-Fields

Another interesting variant of the eigenfaces algorithm was very recently proposed by Gross, Matthews and Baker [20][21] extending the works of Baker, Sim and Kanade [22][23]. It calculates eigen light-fields from collections of face images and uses these light-fields instead of the raw images to span up the subspace. The primary advantage with the eigen light-field approach is its ability to recognize images captured from arbitrary poses and under different illumination conditions, where most other algorithms require training images at every pose to succeed.

Belhumeur and Kriegman have shown that a set of images of an object in a fixed pose, under all possible illumination conditions, is a convex cone in the space of images [24][25]. Using only a small number of training images of each face taken with different lighting directions the shape of the face can be reconstructed by interpolating the missing angles.

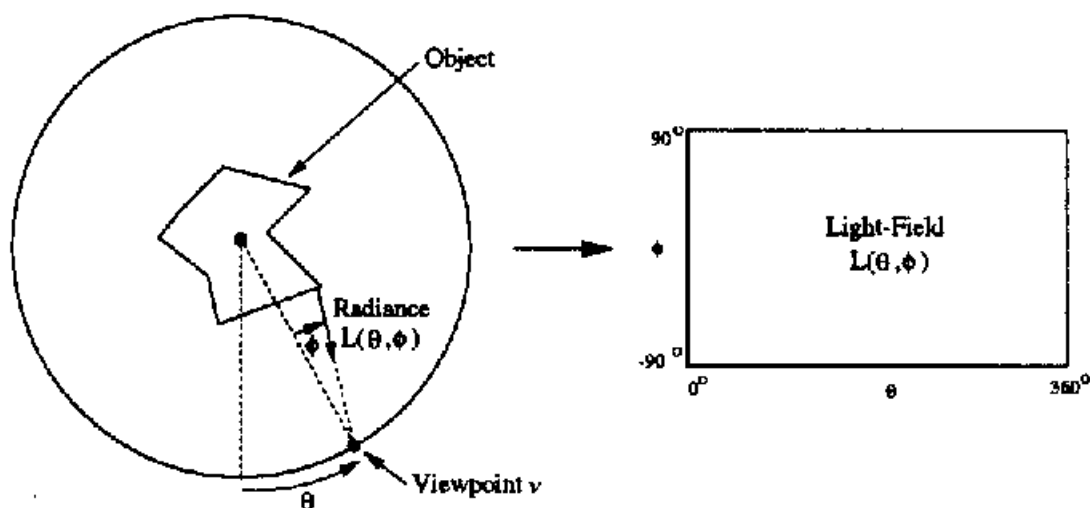


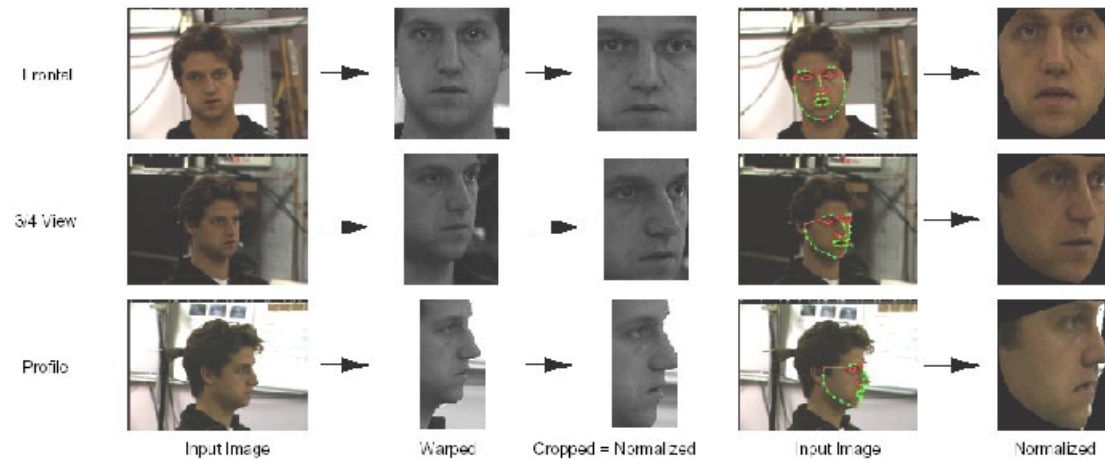
Figure 7: Simplified example of a 2D light-field from a 2D object

The plenoptic function [26] or light-field [27] specifies the radiance of light from an object seen from every position outside the object in every direction. It is typically assumed to be a 5D function, consisting of position (3D) and orientation (2D). It is also sometimes modeled as a function of wavelength, polarization and time.

Assuming no absorption, scattering or emission of light through the air, the light-field can be modeled as being only a 4D function, a 2D direction on a 2D surface. An image of a 2D object (Figure 7) will represent a curve in the light-field, and for 3D objects it will be a surface.

Capturing the complete light-field of an object requires a huge number of images, in contrast to the fact that it is unlikely to expect having more than a few images of the same individual. But the light-field can fortunately be estimated from only a small number of image samples because of the redundancy of radiation from Lambertian objects. Faces can be considered to be Lambertian because their surfaces reflect light diffusely. A light-field captured by a finite number of images can be considered as being occluded, where the object is only visible from a few angles. Leonardis and Bischof describe a method of dealing with occlusions in eigenspace and finding a minimum square solution [28][29]. Once the light-field has been estimated it can be

used for face recognition across pose or to interpolate new images of the same object under different poses. Tests show that the performance improves with the number of training images of the same individual stored in the light-field vector of images, and shows excellent recognition on profile views compared to other algorithms.



**Figure 8: (a) 3-point normalization (left) (b) multi-point normalization (right)**

The running time during training of the eigen light-field algorithm compared to the classical eigenface algorithm is only insignificantly more computational demanding. However, one major difficulty with the eigen light-field algorithm is that one has to know the relative orientation between the camera and the object. Finding the orientation of a face is no easy task and can certainly be a time consuming one. Gross, Matthews and Baker present two methods for estimating an unknown head orientation [21]. First a 3-point normalization using the eye and nose coordinates is presented (Figure 8a). Secondly a multi-point normalization using active appearance models consisting of 39-54 points in the face depending on the pose (Figure 8b).

#### 4.2.5 Evaluation

Results show that eigenfaces methods are robust over a wide range of parameters and produce good recognition rates on various databases [31]. However outside this parameter range the algorithm can breakdown sharply. Results show that eigenfaces are very robust to low resolution images as long as the preprocessing step can extract sufficient features for normalization. They also handle high resolution images very efficiently. It seems to be vital that the preprocessing step is working well. A normalization process can solve rotation issues by aligning both eyes horizontally, scale by adjusting the distance between the eyes and translation by cropping the image. Horizontal and vertical misalignment of only 5% because of difficulties in detecting face features like the eyes have severe effects on the recognition rates. Significant variation in scale, orientation, translation and lightning will also cause it to fail. One has to remember the fact that the reflectance is different for a translated image because of a slightly different viewing angle. An image from the database of face taken from one meter away will therefore not perfectly match a face taken five meters away and zoomed in an appropriate amount.

The choice of distance measure has also proven to affect the performance of face recognition. Euclidean distance (L2 norm) is the most commonly used measure and is computational easy. Yambor, Draper and Beveridge claim that the Mahalanobis distance, which uses the eigenvalues as weighting for the contribution of each axis in the eigenspace, outperforms all the other measures when having a subspace spanned by more than 20 eigenvectors [32][33]. Beveridge, She, Draper and Givens also conclude that PCA with Mahalanobis distance is the best combination [34].

Selection of  $k$ , the number of eigenfaces to keep, is also an important choice because using a low number will fail to capture all the differences in the dataset, while using a high number will be computational demanding. For large databases like the FERET database at least 200 eigenfaces are needed to sufficiently capture global variations like lighting, small scale and pose variations, race and sex [35]. The results may improve by dropping some of the eigenvectors either from the front (lighting) or the back (noise).

A solution to the fundamental problem of handling pose variations seems to be using the new eigen light-field approach, but the normalization process can become time consuming when the orientation between the face and the camera is unknown and has to be estimated. Another solution handling pose variations is having several sets of eigenvectors representing different views. The recognition results are better, but the computational cost is higher.

The low computational cost recognizing faces with the traditional eigenface method comes as a result of a high computational cost training the faces [36]. In the construction of the training set, one can imagine a new face that is not well represented by the eigenfaces calculated from this training set. In this case it becomes necessary to update the training set, which implies an update of the eigenfaces. It is always possible to do a full recalculation of the eigenfaces, but this is a time consuming process. Chandrasekaran, Manjunath, Wang, Winkeler and Zhang of University of California have proposed a method of incremental updating of the eigenspace for images being significantly outside the current object eigenspace [37][38].

<b>Advantages</b>	<b>Drawbacks</b>
<ul style="list-style-type: none"> <li>+ Robust against noise and occlusion</li> <li>+ Robust against illumination, scaling, orientation and translation when face is correctly normalized</li> <li>+ Robust against facial expressions, glasses, facial hair, makeup etc.</li> <li>+ Can handle high resolution images efficiently</li> <li>+ Can handle very low resolution images</li> <li>+ Can handle small training sets</li> <li>+ Fast recognition/Low computational cost</li> </ul>	<ul style="list-style-type: none"> <li>- Removes neighbourhood relationships between pixels</li> <li>- Sensitive to faulty normalization</li> <li>- Sensitive to perspective, viewing angle and head rotation (can be improved using eigen light-fields or other view-based methods)</li> <li>- Sensitive to large variation in illumination and strong facial expressions</li> <li>- Slow training/High computational cost (with large databases)</li> </ul>

### 4.3 Linear Discriminant Analysis (LDA)

#### 4.3.1 Fisherfaces

R. A. Fisher developed Fisher's Linear Discriminant (FLD) [39] in the 1930's but not until recently have Fisher discriminants been utilized for object recognition. Swets and Weng used FLD to cluster images for the purpose of identification in 1996 [40]. Also in 1997, Belhumeur, Hespanha and Kriegman of Yale University used FLD to identify faces, by training and testing with several faces under different lighting [41].

Fisher Linear Discriminant (FLD) analysis, also called Linear Discriminant Analysis (LDA) finds the line that best separates the points. For example, consider two sets of points, coloured green and blue, in two-dimensional space being projected onto a single line. Depending on the direction of the line, the points can either be mixed together (Figure 9a) or be separated (Figure 9b). In terms of face recognition this means grouping images of the same class and separate images of different classes. Images are projected from a  $N$ -dimensional space, where  $N$  is the number of pixels in the image, to a  $M-1$  dimensional space, where  $M$  is the number of classes of images [33][42][43].

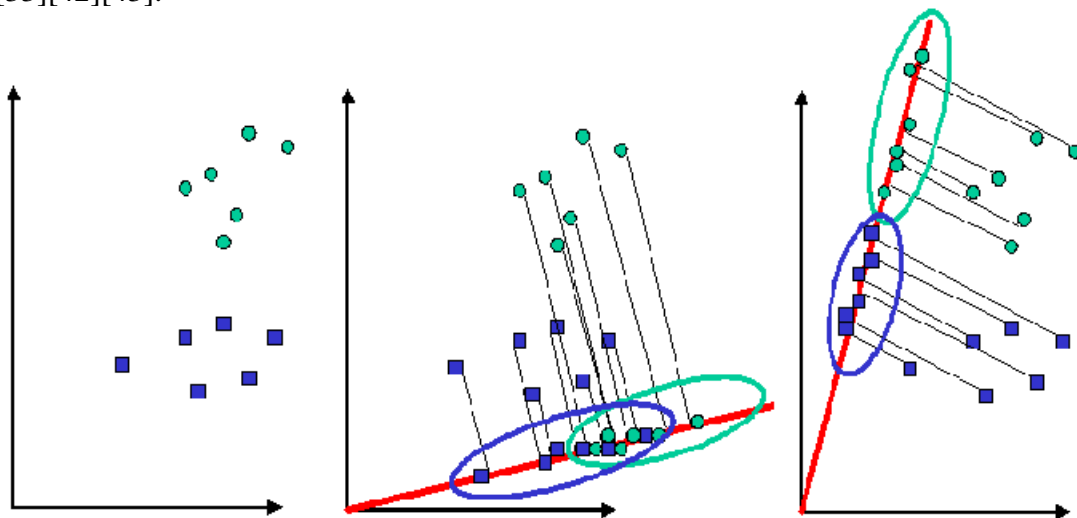


Figure 9: (a) Points in two-dimensional space (b) poor separation (c) good separation

The approach is similar to the eigenface method, which makes use of projection of training images into a subspace. The test images are projected into the same subspace and identified using a similarity measure. What differs is how subspace is calculated. The eigenface method uses PCA for dimensionality reduction, which yields directions that maximize the total scatter across all classes of images. This projection is the best for reconstruction of images from a low-dimensional basis. However, the method does not make use of between-class scatter between classes of face images belonging to the same individual. A PCA projection may not create an optimal discrimination for different classes.

The LDA method, which creates an optimal projection of the dataset, maximizes the ratio of the determinant of the between-class scatter matrix of the projected samples to the determinant of the within-class scatter matrix of the projected samples. The within-class scatter matrix, also called intra-personal, represents variations in appearance of the same individual due to different lighting and face expression, while the between-class scatter matrix, also called the extra-personal, represents variations in appearance due to a difference in identity. In this way fisherfaces can project away some variation in lighting and facial expression while maintaining discriminability. [44]

### 4.3.2 LDA in the Fourier Domain

Akamatsu, Sasaki and Suenaga applied LDA to the Fourier spectrum of the intensity image [30]. The results reported by the authors showed that LDA in the Fourier domain is significantly more robust to variations in lighting than the LDA applied directly to the intensity images. However the computational complexity of this method is significantly greater than the classical fisherface method due to the computation of the Fourier spectrum.

### 4.3.3 Fisher Light-Fields

After pose variation, the next most significant factor affecting the appearance of faces is illumination. Only a few approaches have been proposed to handle both pose and illumination variation at the same time. These include methods creating full 3D head models requiring a large number of training images. This algorithm can however use any number of training images captured at arbitrary poses and under arbitrary illuminations. The test image used for probing can also be of arbitrary pose and illumination. The matching process is as usual performed using a distance measure between the test image and the training images.

Gross, Matthews and Baker have been writing about eigen light-fields, and have now extended their works by writing about fisher light-fields [45]. The fisher light-fields method is a natural extension to fisherfaces, just as eigen light-fields is a natural extension to eigenfaces. Experiments show that as with the eigen light-fields method the number of training images is important.

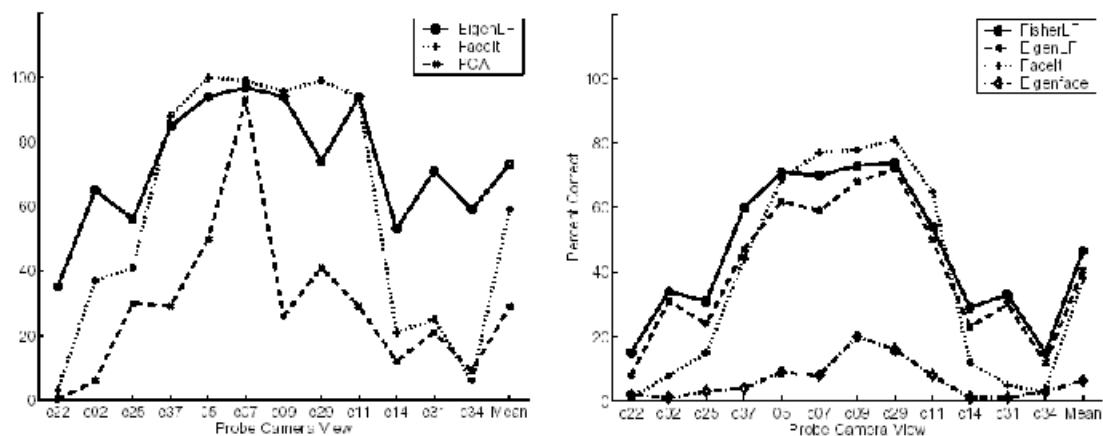


Figure 10: Algorithm comparison (a) across pose (left) (b) across pose and illumination (right)



Tests which were conducted using images from the PIE face database showed on average better performance with fisher light-fields compared to eigen light-fields. Recognition across pose (Figure 10a) and across both pose and illumination (Figure 10b) was compared for light-fields compared to the commercial available FaceIt software and the classical eigenfaces method. Light-field methods showed better recognition rates compared to FaceIt when having large pose variations. The classical eigenface method performs poorly both across poses and illumination in all these tests.

#### 4.3.4 Evaluation

The fisherface method is very similar to the eigenface method, but with improvement in better classification of face images by using interclass and intraclass relationships to separate them. With LDA it is possible to classify the training set to deal with different people and different facial expressions. The accuracy for handling facial expressions has shown to be better than the eigenfaces method.

The fisherfaces method is quite insensitive to large variations in lighting direction and facial expression. Compared to the eigenface method this algorithm is more complex, something which increases the computational requirements, but show lower error rates. Besides, due to the need of better classification, the dimension of the projection in face space is not as compact as in the eigenfaces approach. This results in larger storage of the faces and more processing time in recognition.

Another drawback comes from the fact that the fisherface method uses particular class information and therefore is recommended to have many images per class in the training process. On the other hand, having many images belonging to the same class can make the recognition system suffer from a lack of generalization resulting in a lower recognition rate. In general the algorithm is performing very well, but cannot always work. In general it fails when the between class scatter is inherently greater than the within class scatter. [46]

<b>Advantages</b>	<b>Drawbacks</b>
<ul style="list-style-type: none"> <li>+ Robust against noise and occlusion</li> <li>+ Robust against illumination, scaling, orientation and translation when face is correctly normalized</li> <li>+ Robust against facial expressions, glasses, facial hair, makeup etc.</li> <li>+ Can handle high resolution images efficiently</li> <li>+ Can handle very low resolution images</li> <li>+ Fast recognition/Low computational cost</li> </ul>	<ul style="list-style-type: none"> <li>- Removes neighbourhood relationships between pixels</li> <li>- Sensitive to faulty normalization</li> <li>- Sensitive to perspective, viewing angle and head rotation (can be improved using fisher light-fields)</li> <li>- Does not handle small training sets well</li> <li>- Slow training/High computational cost (with large databases)</li> </ul>

## 4.4 Independent Component Analysis (ICA)

### 4.4.1 Introduction

Independent Component Analysis (ICA) is a technique for extracting statistically independent variables from a mixture of them [47]. The technique is quite new and has originated from the world of signal processing. A classical example demonstrating the original problem is the cocktail-party problem where two people being in the same room speak simultaneously. Two microphones are placed at different locations recording the mixed conversations. It would be very useful if one could estimate the two original speech signals from the two mixed recordings. Surprisingly it turns out that it is enough to assume that the two speech signals are statistically independent. This is not an unrealistic assumption, but it does not need to be exactly true in practice. ICA can be used to estimate the contribution coefficients from the two signals, which allows us to separate the two original signals from each other. Hyvärinen and Oja have written a good tutorial about ICA which contains more details about the algorithms involved [48].

In a task such as face recognition, much of the important information may be contained in the high-order relationships among the image pixels. Some success has been attained using data-driven face representations based on PCA, such as eigenfaces. PCA is based on the second-order statistics of the image set, and does not address high-order statistical dependencies such as the relationships among three or more pixels. Independent component analysis (ICA) however separates the high-order moments of the input in addition to the second-order moments. ICA thus in some ways provide a more powerful data representation than PCA, as its goal is to provide an independent rather than an uncorrelated image decomposition and representation.

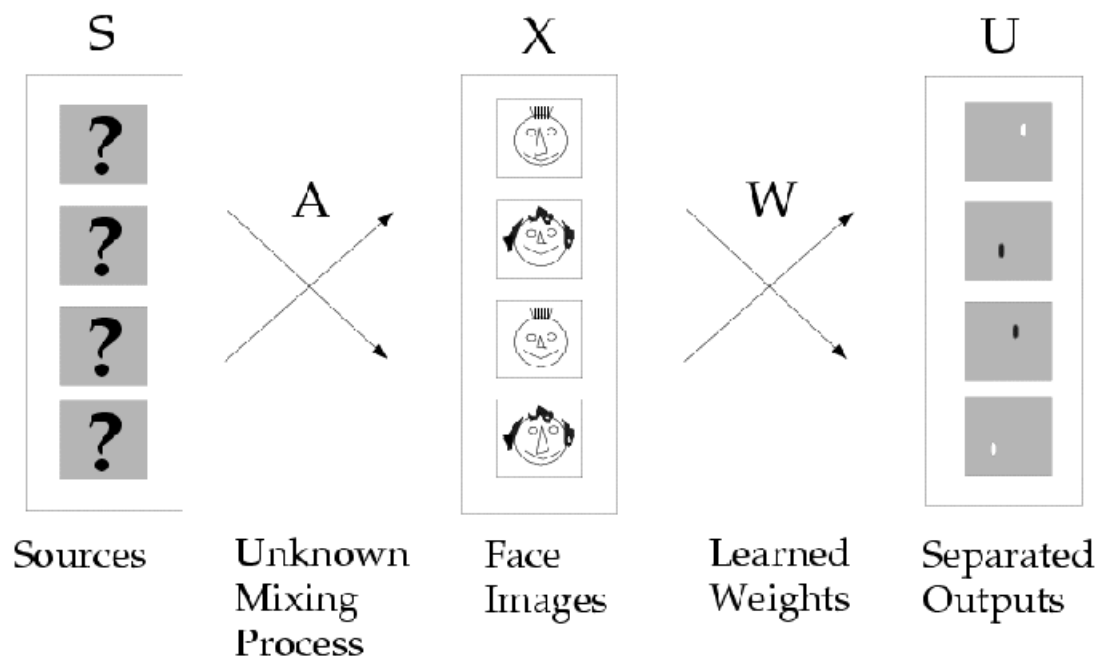
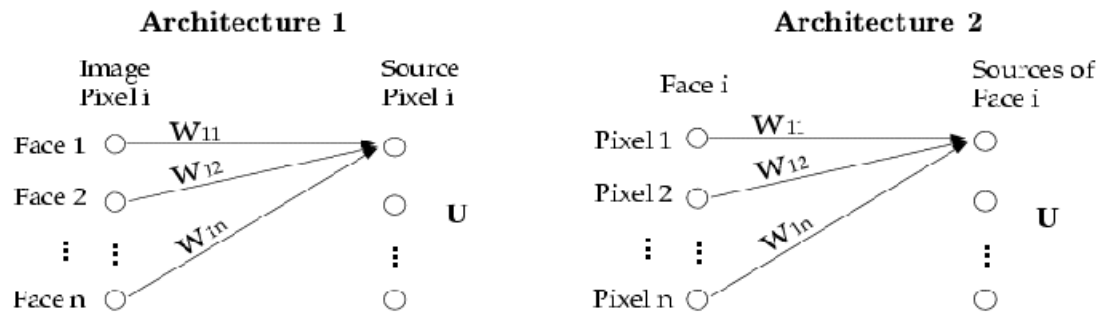


Figure 11: ICA image synthesis model

For finding a set of independent component images, the face images  $X$  are considered to be a linear combination of statistically independent basis images  $S$ , where  $A$  is an unknown mixing matrix. The basis images are recovered by a matrix of learned filters  $W$ , which produces statistically independent outputs  $U$  (Figure 11).



**Figure 12: ICA on images with source separation on (a) face images (b) face pixels**

Bartlett and Sejnowski at University of California have used ICA for face recognition [49][50][51]. Two approaches for recognizing faces across changes in pose were explored using ICA. The first architecture provided a set of statistically independent basis images for the faces that can be viewed as a set of independent facial features (Figure 12a). This corresponds very much to the classical cocktail-party problem performing a blind separation of a mixture of auditory signals. These ICA basis images were spatially local, unlike the PCA basis vectors. The representation consisted of the coefficients for the linear combination of basis images that comprised each face image. The second architecture produced independent coefficients (Figure 12b). This provided a factorial face code, in which the probability of any combination of features can be obtained from the product of their individual probabilities. Classification was performed using nearest neighbour, with similarity measured as the cosine of the angle between representation vectors. Both ICA representations showed better recognition scores than PCA when recognizing faces across sessions, changes in expression, and changes in pose.

#### 4.4.2 Tensorfaces

Natural images are the consequence of multiple factors related to scene structure, illumination and imaging. Multilinear algebra offers a mathematical framework for analyzing collections of images and the factors involved. Vasilescu and Terzopoulos have specifically considered multilinear analysis of collections of face images that combine several modes like different individual, expressions, head poses, lighting conditions etc [52][53]. Their modelling technique is also known as N-mode singular value decomposition (SVD). Tensor decompositions can be used in conjunction with higher order statistics employed in ICA.

### 4.4.3 Evaluation

In 1999 Liu and Wechsler also claimed that ICA produced better results or matched the results that were obtained purely by PCA [54]. This was later in 2001 contradicted by Baek, Draper, Beveridge and She [55]. They showed that PCA outperformed ICA when the distance method is selected to maximize performance. Both experiments were conducted using the FERET database. The most recent contradicting results from 2001 however showed that the differences in recognition rates between PCA and ICA are only minor, and very much depend on how the algorithms in detail are implemented.

Global properties like coloring, width and length are more easily captured by PCA than ICA, since ICA basis vectors are more spatially localized than their PCA counterparts. Recognizing more localized features, like face expressions, may produce significantly different results.

<b>Advantages</b>	<b>Drawbacks</b>
<ul style="list-style-type: none"> <li>+ Considers higher-order relationships</li> <li>+ Robust against noise and occlusion</li> <li>+ Robust against illumination, scaling, orientation and translation when face is correctly normalized</li> <li>+ Robust against facial expressions, glasses, facial hair, makeup etc.</li> <li>+ Fast recognition/Low computational cost</li> </ul>	<ul style="list-style-type: none"> <li>- Removes neighbourhood relationships between pixels</li> <li>- Sensitive to faulty normalization</li> <li>- Sensitive to perspective, viewing angle and head rotation</li> <li>- Slow training/High computational cost (with large databases)</li> </ul>

## 4.5 Wavelets

### 4.5.1 Introduction

Wavelets represent an approach to decomposing complex signals into sums of basis functions. In this respect they are similar to Fourier decomposition approaches, but they have an important difference. Fourier functions are localized in frequency but not in space, in the sense that they isolate frequencies, but not isolated occurrences of those frequencies. This means that small changes in a Fourier transform will produce changes everywhere in time domain. Wavelets are local in both time by translations and frequency by dilations. Because of this they are able to analyze data at different scales or resolutions much better than simple sine and cosines can. To understand this note that modelling a spike in a function, a noise dot for example, with a sum of infinite functions will be hard because of its strict locality, while functions that are already local will be naturally suited to the task. Sharp spikes and discontinuities normally take fewer wavelet bases to represent than if sine-cosine basis functions are used.

### 4.5.2 Gabor Wavelets

Physiological studies have found simple cells in human visual cortex which are selectively tuned to orientation as well as to spatial frequency. The response of these simple cells can be approximated by 2D Gabor filters [56]. Gabor functions were first proposed by Dennis Gabor as a tool for 1D signal detection in noise [57].

Rediscovered and generalized to 2D Gabor wavelet representation for computer vision was pioneered by Daugman in 1980 [58]. Manjunath, Chellapalla and Malsburg have developed a face recognition system based on this representation [59]. This work has continued with elastic bunch graph matching of coefficients from Gabor filter responses by Wiskott, Fellous, Krüger and Malsburg [60] and the dynamic link architecture by Lades, Vorbrüggen, Buhmann, Lange, Malsburg, Würtz and Konen [61]. Gabor filters are now being used extensively in various computer vision applications.

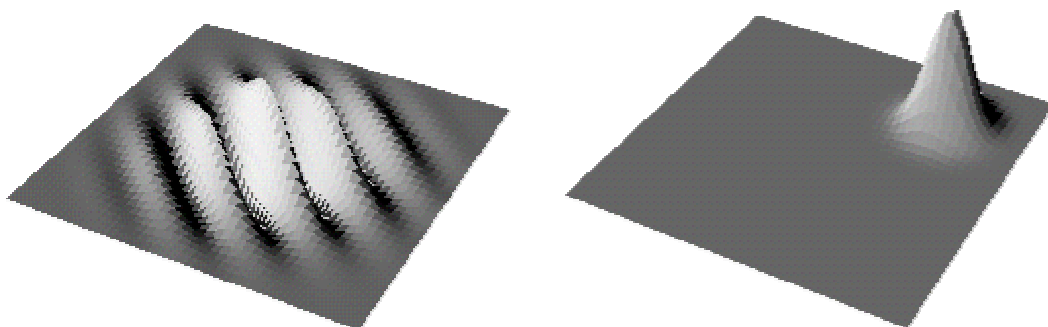


Figure 13: 3D representation of a gabor wavelet in (a) space domain (b) frequency domain

2D Gabor functions are similar to enhancing edge contours, as well as valleys and ridge contours of an image. The most important face features like eyes, mouth and nose edges are enhanced, as well as moles, dimples and scars. Gabor wavelets have the shape of plane waves restricted by a Gaussian envelope function (Figure 13). An image can be represented by Gabor wavelet responses by convolving Gabor filters of

different spatial frequency/size and orientation/rotation (Figure 14). The figure below shows an ensemble of gabor wavelets consisting of 3 different frequencies and 8 different orientations and their coverage of spatial frequency plane.

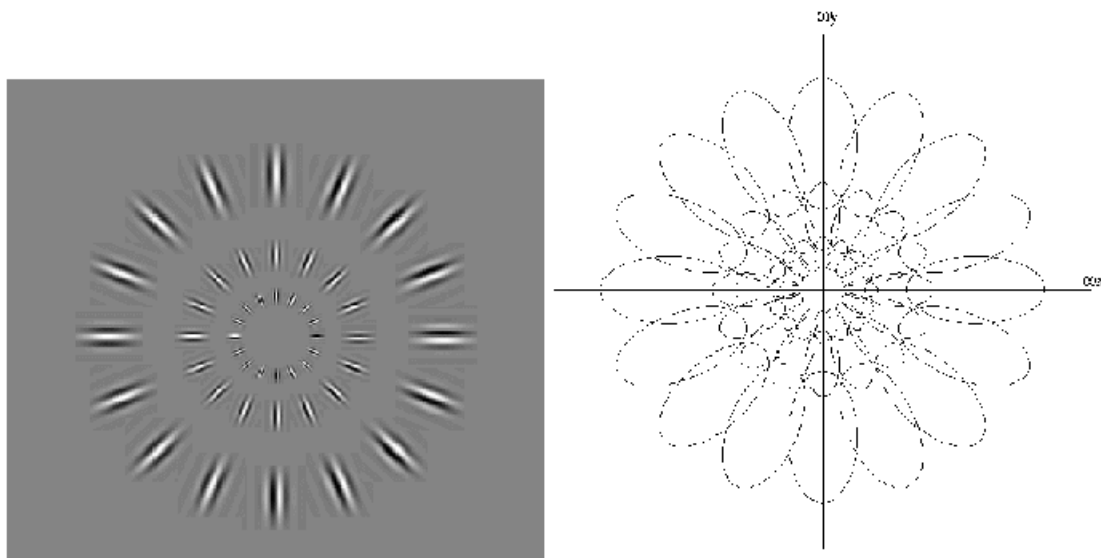


Figure 14: (a) Ensemble of gabor wavelets (b) Coverage of spatial frequency plane

Both amplitude and phase are captured describing spatial frequency structure and spatial relations. The set of convolution coefficients for kernels of different orientations and frequencies at one image pixel is called a jet (Figure 15).

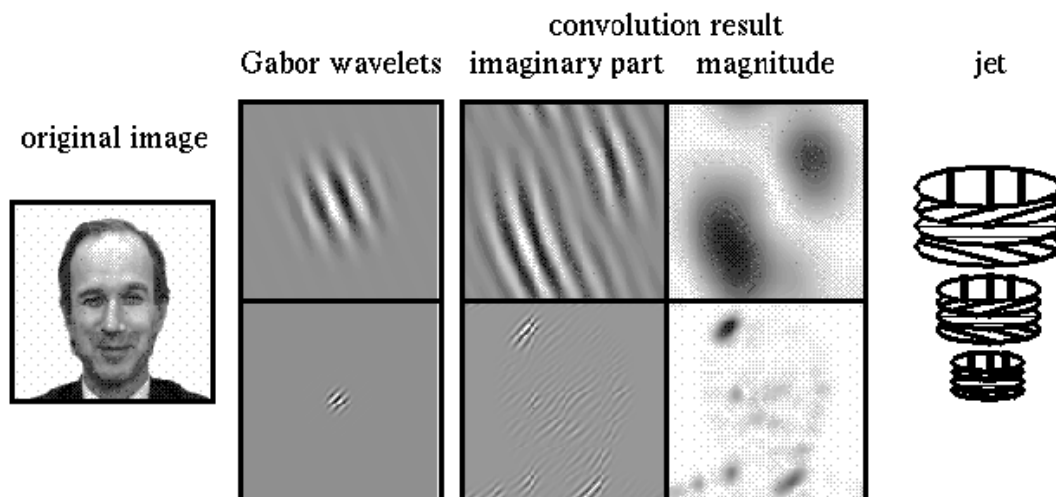


Figure 15: (a) Original image (b) Gabor wavelets (c) Convolution result (d) Jet

The easiest technique for comparing Gabor jets of different images is using a regular spaced grid covering the images to be recognized. Comparing jets can be very difficult. Due to phase rotation, jets taken from image points only a few pixels away from each other have very different coefficients, even if they are representing almost the same local feature. We can therefore either ignore phase or compensate for its variation explicitly. Using the phase information is required to discriminate between

two patterns with similar amplitudes. Since phase varies so quickly it also provides means for accurate jet localization in images.

### 4.5.3 Elastic Bunch Graph Matching

Different human faces have the same geometrical structure and can therefore be defined as labelled graphs. Since we want to recognize faces from different views, the nodes of the graphs consistently refer to particular fiducial points, such as eyes, mouth, the tip of the nose and other contour points (Figure 16). Graphs for different head pose differ in geometry and local features. Although the fiducial points refer to corresponding object locations, some may be occluded, and jets as well as distances vary due to rotation in depth. To be able to compare graphs from different poses pointers have to be established to associate corresponding nodes in the different graphs.

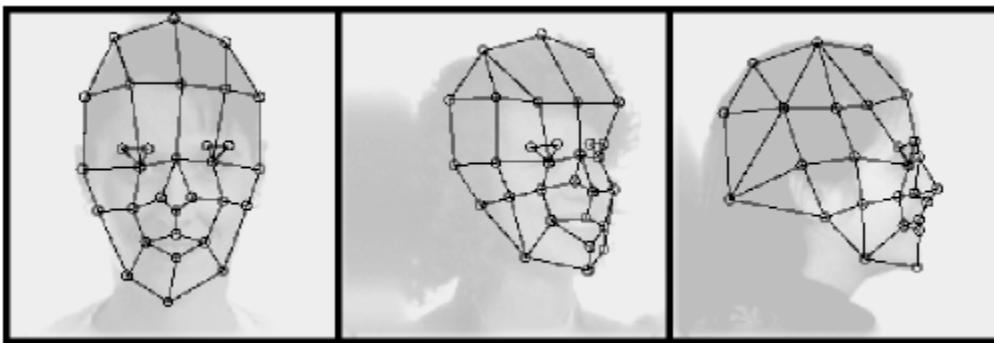


Figure 16: Graphs for faces in different views

A system that has to deal with large galleries can not afford to match each model to a new face separately. A common approach is to combine different models into a face bunch graph, which is a general representation rather than models of individual faces. This representation covers a wide range of possible variations in the appearance of faces, such as differently shaped eyes, nose and mouths, different types of beards, variations due to gender, age, race etc. The Face Bunch Graph (FBG) has a stack-like structure and combines graphs of individual sample faces (Figure 17).

It is crucial that the individual graphs all have the same structure and that the nodes refer to the same fiducial points. Every fiducial point or graph node consists of jets that are bundled together in a bunch, from which one can select any jet as an alternative description. The left eye bunch might contain representations of a male eye, a female eye, closed eye and open eye etc. Each fiducial point is represented by such a set of alternatives and from each bunch any jet can be selected independently of the jets selected from the other bunches. That provides full combinatorial power of this representation and makes it so general even if constituted from few graphs only. During the location of fiducial points in a face not seen before the best fitting jets, called the local experts, are selected from the face bunch graph and one is dedicated to each fiducial point.

Initially, when the face bunch graph consists of only a few faces, it is necessary to manually review and correct the resulting matches, but once the face bunch graph is rich enough one can rely on the matching and generate large galleries automatically.

Approximately 70 graphs are needed to make the face bunch graph rich enough. The first step in creating the face bunch graph consists of a normalization step estimating the position and size of the face in the original image, so that it can be scaled and cut to standard size. The second stage takes this image as input and extracts a precise image graph appropriate for face recognition purposes. It can start with a little uncertainty as to position and size of the head, but has to extract the face graph with high precision [62].

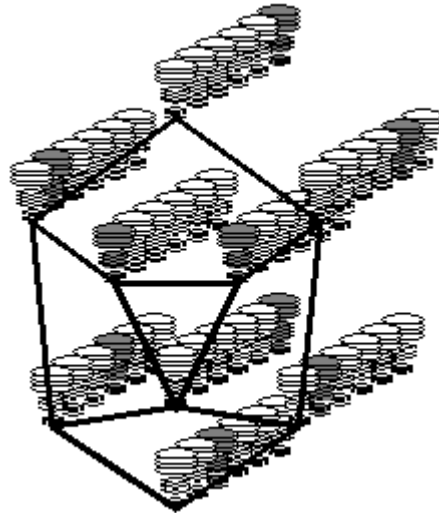


Figure 17: Face bunch graph with local experts at each fiducial point marked grey

Kepekci has recently proposed a method of selecting high-energy peaks of the Gabor wavelet response instead of using predefined graph nodes as in elastic graph matching [63]. This reduces computational complexity and also improves the performance in the presence of occlusions. Hjelms reports of 85% recognition on the ORL database [64][65].

#### 4.5.4 Gabor Fisher Classifier (GFC)

Liu at University of Missouri and Wechsler at George Mason University have applied an enhanced Fisher Discrimination Model (EFM) to the Gabor feature vector [66][67]. The dimensionality of the vector space is reduced in order to derive a low-dimensional feature representation with enhanced discrimination power. The GFC method is robust to illumination and facial expression variability and they report about excellent performance on the FERET database compared against other methods using Gabor wavelets, eigenfaces, fisherfaces, and a combination of Gabor and eigenfaces.

#### 4.5.5 Evaluation

Gabor wavelets are chosen for their robustness as a data format and for their biological relevance. One of the main motivations for using such feature based methods is that representation of face images in this way becomes very compact and this lowers the computational cost. This fact especially gains importance when there is a huge database.



Since Gabor responses are DC-free they provide robustness against varying brightness in the image. Robustness against varying contrast can be obtained by normalizing the jets. The limited localization in space and frequency yields certain amount of robustness against translation, distortion, rotation and scaling. Face Bunch Graphs represent a good data structure for storing the extracted features. A simple graph consisting of only nine nodes and six jets can theoretically represent  $6^9$  or about as many as ten million different faces.

Finding the locations and corresponding values of the fiducial points in a face image is extremely critical for the performance of the recognition system. However, some of the most successful face recognition methods are based on graph matching of Gabor filter responses. Disadvantages are the graph matching complexity [68], manual location of training graphs and overall execution time.

<b>Advantages</b>	<b>Drawbacks</b>
<ul style="list-style-type: none"> <li>+ Saves neighbourhood relationships between pixels</li> <li>+ Robust against illumination, scaling, orientation and translation when face is correctly normalized</li> <li>+ Robust against noise</li> <li>+ Robust against translation, rotation and scaling</li> <li>+ Easy to update</li> <li>+ Fast recognition/Low computational cost</li> </ul>	<ul style="list-style-type: none"> <li>- Sensitive to faulty normalization</li> <li>- Sensitive to facial expressions, glasses, facial hair, makeup etc. (can be improved using elastic bunch graph matching)</li> <li>- Sensitive to occlusion (can be improved using high energy feature points as graph nodes)</li> <li>- Sensitive to perspective, viewing angle and head rotation (can be improved using elastic bunch graph matching)</li> <li>- Graph matching complexity</li> <li>- Slow training/High computational cost (with large databases)</li> </ul>

## 4.6 Hidden Markov Models (HMM)

### 4.6.1 Introduction

The use of hidden Markov models is a powerful statistical technique that has been applied to many subject areas, from predicting political crises to the reconstruction of DNA and the recognition of speech. The September 1964 issue of Scientific American illustrated a Markov chain by showing two containers with numbered balls in them. Numbered slips of paper associated with the balls were drawn repeatedly, with replacement, from a hat. The ball associated with the number drawn was transferred to the other container than the one it was in. Initially all the balls were in the first container, and gradually this declined exponentially until it contained only half of the balls. This modelled the physical process of allowing two separate chambers, containing a gas at different levels of pressure to be connected. One basic feature with the Markov process is that it involves probability. In addition to a random event the final result also depends on some kind of system memory, described by the number of balls in the first container.

A hidden Markov model consists of two interrelated processes. First an underlying, unobservable Markov chain with a finite number of states ( $N$ ), a state transition probability matrix ( $A$ ) and an initial state probability distribution ( $\pi$ ). Transition probability is the probability that the system will change its state from one turn to the next. Second a set of probability density functions ( $B$ ) associated with each state. Using shorthand notation a discrete hidden Markov model can be defined as  $\lambda = (N, A, B, \pi)$ . In practice the state sequence is unknown (hidden) and cannot be evaluated. However, the likelihood can be evaluated by summing over all the possible state sequences. The key attraction of HMM is that there is a simple procedure for finding the parameters  $\lambda$  called Baum-Welch re-estimation.

In order to use HMM for recognition, an observation sequence is obtained from the test signal and then the likelihood of each HMM generating this signal is computed. The HMM which has the highest likelihood then identifies the test signal. Finding the state sequence which maximizes the probability of an observation is done using the Viterbi algorithm, which is a simple dynamic programming optimization procedure. More details describing all the technical details concerning the algorithms used can be found in a great tutorial written by Rabiner [69].

One pioneer of using hidden Markov models for face recognition was Samaria at Trinity College starting in 1994 [70][71]. Nefian and Hayes at Georgia Institute of Technology have written several papers on pseudo 2D HMM [72][73][74][75][76] and some on embedded HMM [77][78]. Eickeler, Müller and Rigoll have written about how to get high performance using pseudo 2D HMM [79][80][81]. Some attempts have also been made by Othman and Aboulnasr on 2D HMM [82][83].

### 4.6.2 One-Dimensional HMM

HMM has been extensively used for speech recognition, where data is naturally one-dimensional along the time axis. The equivalent fully-connected two-dimensional HMM would lead to a very high computational cost problem. Samaria has proposed using the 1D continuous HMM for face recognition [71]. For a frontal face the states

of the Markov model include hair, forehead, eyes, nose and mouth, each representing a state. These states always occur in the same order, from top to bottom, even if faces undergo small rotations in the image plane. Each facial region will be assigned to a state, in a left-to-right one dimensional hidden Markov model (Figure 18). Only transitions between adjacent states in a top to bottom manner are allowed.

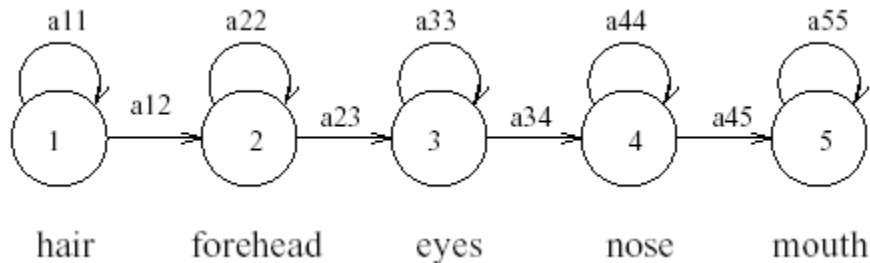


Figure 18: Left-to-right states of a one-dimensional HMM

An observation sequence is generated from a face image ( $X \times Y$ ) using a sampling window ( $M \times L$ ) with overlap (Figure 19). The observation sequence is composed of vectors that represent the consecutive horizontal strips, where each vector contains the pixel values from the associated strips. The goal of the training stage is to optimize the hidden Markov model parameters to best describe the observations. This is done by maximizing the probability of the observed sequence given a set of variable parameters. Recognition is done by matching the test image against each of the trained models. To do this the image is converted to an observation sequence and then model likelihoods for all database images are computed. The model with the highest likelihood reveals the identity of the unknown face.

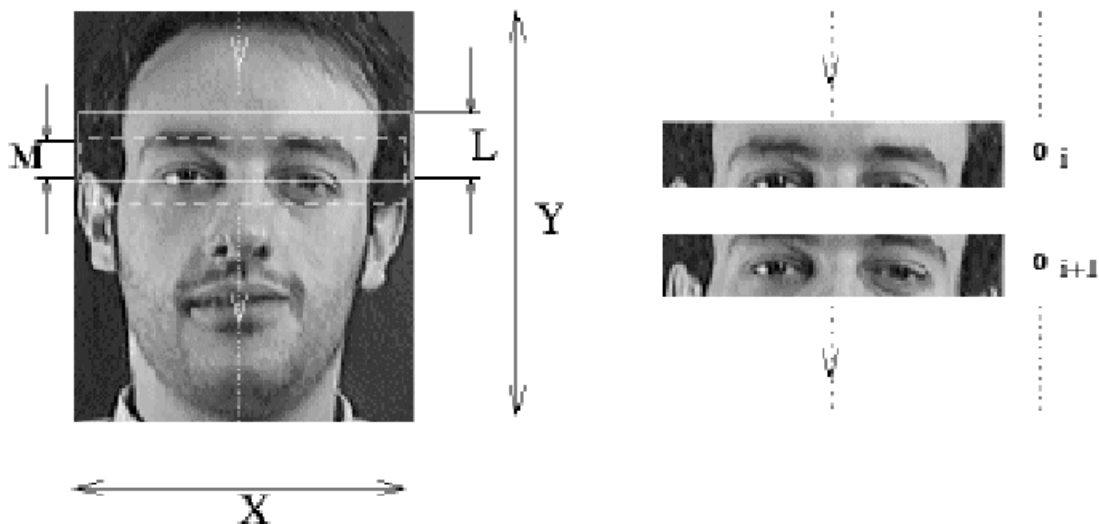


Figure 19: Image sampling technique for one-dimensional HMM

#### 4.6.3 Pseudo and Embedded Two-Dimensional HMM

A more flexible HMM, that allows for shifts in both horizontal and vertical directions, is obtained by using a pseudo two-dimensional HMM. It has been designed specifically to deal with two-dimensional signals and has recently been proposed for face recognition applications. The structure is not fully connected in two-dimensions,

hence it is pseudo two-dimensional. States are linked as in a one-dimensional HMM to form vertical superstates. Each superstate in the one-dimensional HMM is represented by an embedded one-dimensional HMM (Figure 20).

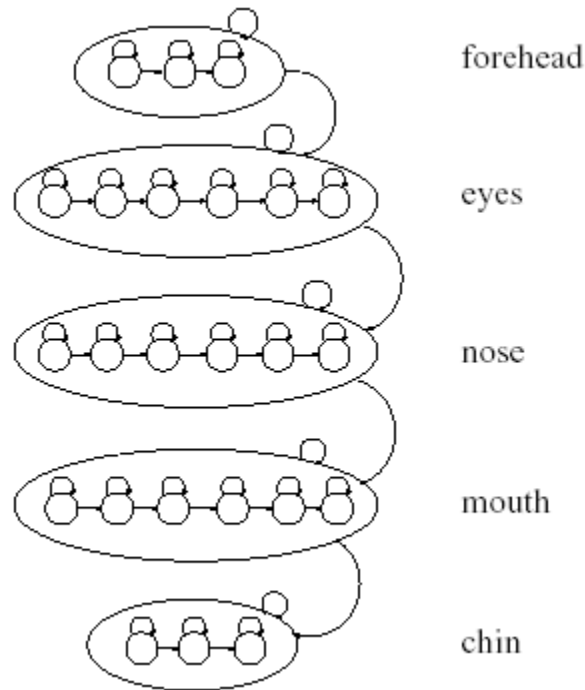


Figure 20: States of a pseudo two-dimensional HMM

Samaria introduced an equivalent one-dimensional HMM and used it for face recognition [71]. The observation sequence is generated by letting a window ( $P \times L$ ) scan the image ( $X \times Y$ ) from left to right, and top to bottom (Figure 21). Each sample overlaps other samples both in horizontal ( $P$ ) and vertical ( $M$ ) direction. The intensities of the pixels inside each block were used as observation vectors.

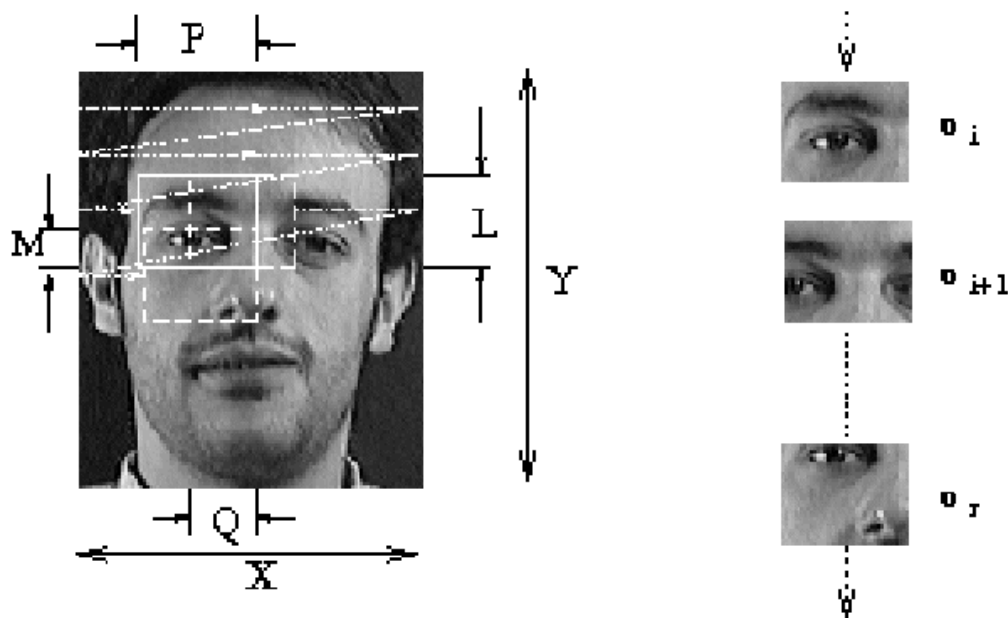
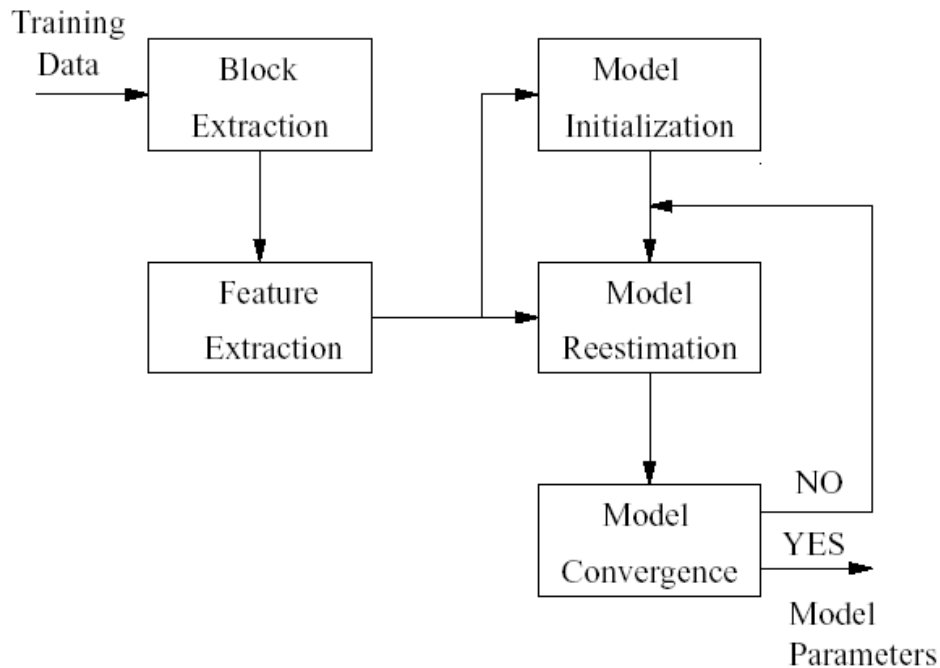


Figure 21: Image sampling technique for pseudo two-dimensional HMM

After extracting blocks from each image in the training set, the observation vectors are obtained to train each of the HMMs. For face recognition each individual in the database is represented by one HMM face model. A set of images representing different instances of the same face are used to train each HMM.



**Figure 22: HMM training scheme**

The general HMM training scheme (Figure 22) is a variant of the K-means iterative procedure for clustering data. First the initial parameter values are computed iteratively using the training data and the prototype model. The goal of this stage is to find a good estimate for the observation probability (B). Good initial estimates of the parameters are essential for rapid and proper convergence to the global maximum of the likelihood function. On the first cycle the data is uniformly segmented, matched with each model state and the initial model parameters are extracted. On successive cycles the set of training observation cycles are segmented into states using the Viterbi algorithm. The result of segmenting each of the training sequences, for each of the N states, is a maximum likelihood estimate of the set of observations that occur within each state according to the current model.

The model parameters are re-estimated using the Baum-Welch re-estimation procedure. This procedure adjusts the model parameters so as to maximize the probability of observing the training data, given each corresponding model. The resulting model is then compared to the previous model by computing a distance score that reflects the statistical similarity of the HMMs. If the model distance score exceeds a threshold then the old model is replaced by the new model and the training loop is repeated. If the model distance score falls below the threshold, then model convergence is assumed and the final parameters are saved.

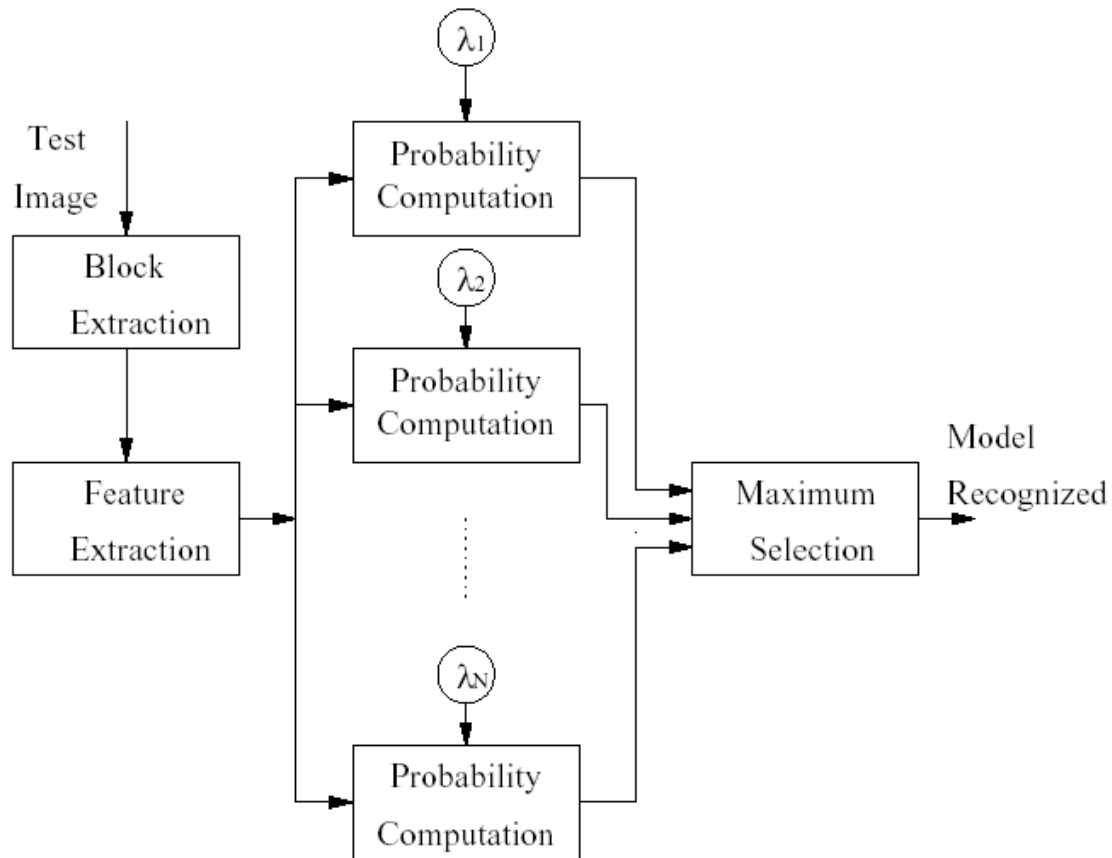


Figure 23: HMM recognition scheme

The face recognition begins by looking within each rectangular window in the test image, extracting observation vectors. After extracting the observation vectors as in the training phase, the probability of the given observation sequence given each face model is computed using a simple Viterbi recognizer. The model with the highest likelihood is selected and this model reveals the identity of the unknown face.

#### 4.6.4 Evaluation

HMM-based methods have shown better performances compared to the traditional eigenfaces method. Error rates of about 5% were reported when pseudo 2D HMM was used compared to about 10% with eigenfaces on the same dataset [71]. The 1D HMM had an error rate of 13% in the same experiment.

The original pseudo 2D HMM uses pixel intensities as input feature vectors. Pixels however do not represent robust features, being very sensitive to image noise as well as image rotation, shift and changes in illumination. Using them is also computational expensive both for training and recognition because of the large dimensions on the feature vectors. This can be critical for a face recognition system that operates on a large database or in real-time systems. Investigations have been made towards using feature vectors containing coefficients from low frequencies using 2D Discrete Cosine Transform (DCT) [73] applied to each observation block. Significant improvements have been attained using DCT coefficients instead of pixel values. One useful property is that it allows recognition in the JPEG and MPEG domain because these

standards use image compression based on DCT. The extraction of DCT coefficients are simply recovered with entropy decoding. This is why pseudo 2D HMM recently was suggested for MPEG-7 v2 standard [84]. Another approach is using Karhunen Loeve Transform (KLT) [74], which also has the necessary feature compression properties.

The Baum-Welch algorithm, which is used for the training of the HMM for each person, provides the HMM parameters corresponding to a local maximum of the likelihood function depending on the initial model parameters. It is therefore very important to use a good initial model for the training. Also as much training data as possible is needed in the estimation of hidden Markov model parameters, to estimate good models for recognition. Block overlap helps in providing higher statistical resolution. However large overlap results in increasing the computational load and memory requirements for all parts of the system. Varying overlap and block size can improve recognition performance.

In order to make the system more tolerant to orientation changes, individual models will have to be trained for views of the same subject at different orientations to the camera. Test images will be matched against models of different subjects and head orientations. It has been shown that at least five models corresponding to different face views are needed for a good face representation under a large range of orientations [13].

The time required by the recognition system is critical. It is a function of the size of the database. Recognition time must be less than the time between two consecutive occurrences of people in a scene. Depending on the parameterization used the Viterbi algorithm can require a large number of calculations. This implies that sometimes the algorithm runs slowly.

One major advantage with HMM is that it provides methods for incremental learning of new classes. This means that new faces can be added to the database without recomputing the representations of all other learned faces.

<b>Advantages</b>	<b>Drawbacks</b>
<ul style="list-style-type: none"> <li>+ Robust against scaling, orientation and translation when face is correctly normalized</li> <li>+ Robust against illumination if training data has different lighting conditions</li> <li>+ Robust against facial expressions, glasses, facial hair, makeup etc.</li> <li>+ Easy to update</li> </ul>	<ul style="list-style-type: none"> <li>- Sensitive to faulty normalization</li> <li>- Sensitive to occlusion</li> <li>- Sensitive to perspective, viewing angle and head rotation (can be improved training models for different views)</li> <li>- Slow training and recognition/High computational cost (can be improved using DCT or KLT feature vectors)</li> </ul>

## 4.7 Neural Networks

### 4.7.1 Introduction

Recognition of visual objects is performed effortlessly in our everyday life by humans. A previously seen face is easily recognized regardless of various transformations like change in size and position. It is known that humans process a natural image in under 150 ms [85]. The brain thus performs these tasks at very high speed. Neural networks are attempts to create face recognition systems that are based on the way humans detect and recognize faces.

### 4.7.2 Multi-Layered Feed-Forward Networks

The multi-layer perceptron (MLP) neural network is a good tool for classification purposes. It can approximate almost any regularity between its input and its output. The weights are adjusted by supervised training procedure called back-propagation (BP). Back-propagation is a kind of gradient descent method, which searches for an acceptable local minimum in order to achieve minimal error. Error is defined as the root mean square of differences between real and desired outputs from the neural network.

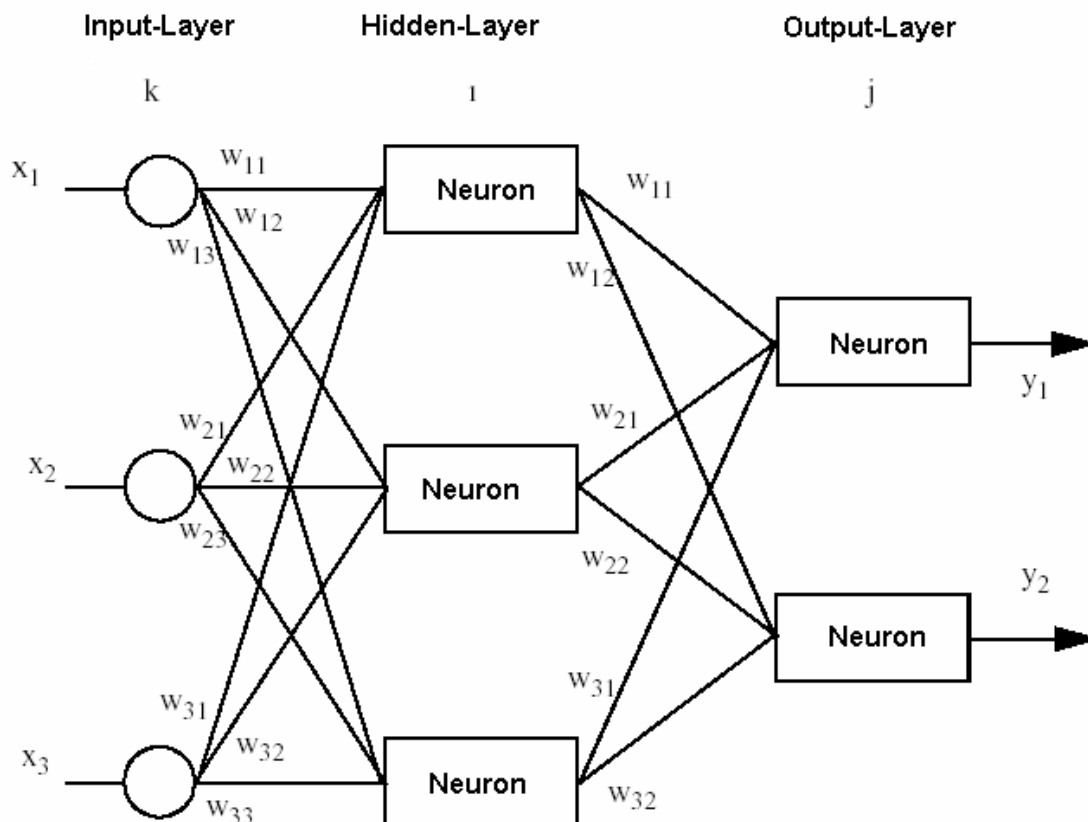


Figure 24: Feed-forward neural network

A typical architecture for a feed-forward network has a number of layers following each other one by one (Figure 24). We have an input layer ( $k$ ) consisting of input nodes and an output layer ( $j$ ) consisting of output nodes. The input nodes are



connected to the output nodes via one or more hidden layers (i) (multilayered). The nodes in the network are connected together, and each of the links has a weight associated with itself. The output value from a node is a weighted sum of all the input values to the node. By changing the different weights of the input values we can adjust the influence from different input nodes. For face recognition the input nodes will typically correspond to image pixel values from the test image to be recognized. The output layer will correspond to classes or individuals in the database. Each unit in the output layer can be trained to respond with +1 for a matching class and -1 for all others. In practice real outputs are not exactly +1 or -1, but vary in the range between these values. The closer the values of the neural network get towards the ideal, the more confidence there is towards the decision being right. Recognition is done by finding the output neuron with the maximal value. Then a threshold algorithm can be applied to reject or confirm the decision.

Experiments have also been made with ensembles of networks where each class in the database has its own neural network [86][87]. The output layer is then trained to give +1 for own person and -1 for other persons. An aggregate output consisting of outputs from all the MLP networks are then considered in the same manner as when having only one MLP and threshold rules can be applied as normal. Huang, Zhou, Zhang and Chen describe a method of pose invariant face recognition using ensembles of networks [88]. They show that the accuracy of ensembles of networks can be higher than single neural networks.

Often even a simple network can be very complex and difficult to train [89]. A typical image recognition network requires as many input nodes as there are pixels in the image. Cottrell and Fleming used two MLP networks working together [90]. The first one operates in an auto-association mode and extracts features for the second network, which operates in the more common classification mode. In this way the hidden layer output constitutes a compressed version of the input image and can be used as input to the classification network. Cottrell and Fleming also showed that a neural network using this design was not any better than an eigenface approach.

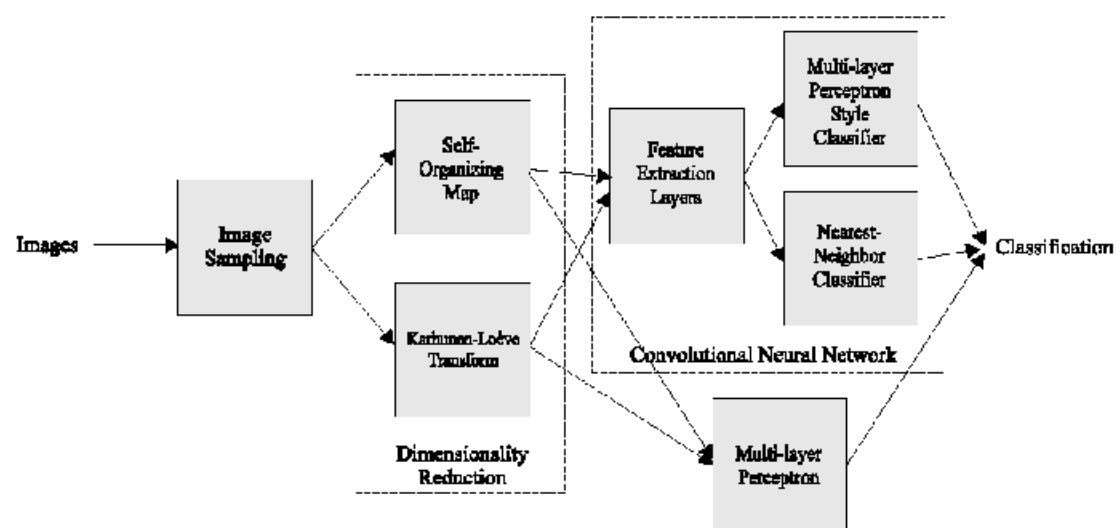


Figure 25: Neural network face recognition system

Maybe one of the more successful face recognition with neural networks is a result of the recent work of Lawrence, Giles, Tsoi and Back at NEC Research Institute. It combines local image sampling, a self organizing map (SOM) neural network and a convolutional neural network [91][92] (Figure 25). SOM was introduced by Kohonen [94] and is an unsupervised learning process which learns the distribution of a set of patterns without having any class information. A pattern is projected from an input space to a position in the map and information is thereby coded as the location of an activated node. Unlike most other classification or clustering techniques SOM preserves the topological ordering of classes. This feature makes it useful in classification of data which includes a large number of classes. Experiments were also made concerning using KLT instead of SOM for dimensionality reduction. A convolutional neural network was trained and compared to a standard MLP network. A major disadvantage is that SOM as well as the convolutional network needs a considerable time to be trained.

### 4.7.3 Radial Basis Function (RBF) Networks

RBF neural networks have recently attracted extensive research interests in the community of neural networks. Their learning speed is fast because of local-tuned neurons and they have a more compact topology than other neural networks. The RBF network is a two-layer feed-forward network, with a supervised layer from the hidden to the output nodes, and an unsupervised layer from the input to the hidden. Gaussian functions for each of the hidden units simulate the effect of overlapping and locally tuned receptive fields.

Howell and Buxton at University of Sussex have written several articles about using RBF networks for face recognition tasks [95][96][97][98][99]. They experimented with using either difference of Gaussian (DoG) or Gabor wavelets as input to the network. Using Gabor wavelets as input gave the best recognition results allowing different scales and orientations to be tailored to the task at hand.

Some approaches have been made towards reducing the input size to the RBF network. Er, Wu and Lu at Nanyang Technological University [100][101] and Feitosa at University of Rio de Janeiro [102] have proposed using PCA and LDA eigenvectors as input to the RBF network to reduce dimensionality. Huang, Law and Cheung at Zhong Shan University have written an article about using ICA together with RBF networks [103]. Results show that these approaches converge faster than the conventional RBF during training, and also outperform its generalization abilities. Gutta and Wechsler have demonstrated the capability of RBF networks to handle large databases, like FERET [104].

### 4.7.4 Dynamic Link Architecture (DLA)

In dynamic link matching the image and all the models are represented by layers of neurons labelled by jets as local features. Jets are vectors of Gabor wavelet components. In each layer neural activity dynamics generates one small moving blob of activity. If a model is similar in feature distribution to an image, its initial connectivity matrix will connect corresponding points having high feature similarity. Blobs in the image and the model tend to align and synchronize by simultaneously activating and generating correlations between corresponding regions. These

correlations are used to restructure and improve the connectivity matrix. This provides translational invariance as well as robustness against distortions. The main concerns with DLA is processing time and its inability to handle large size and orientation changes. Wiskott and Malsburg have written a good article which describes DLA and its algorithms in detail [105].

#### 4.7.5 Evaluation

Neural networks have been used in many recognition tasks and have achieved high recognition rates for limited datasets. The representation of the given input to the network and the training phase is crucial for the results of the face recognition. The representation of the given input, the hidden layer network, the coupling between the network components and the transfer function are vital elements deciding the functionality and the performance of the neural network face recognition system. Achieved recognition results are dependent on the database size and the number of pictures per person. The training time is growing with the number of pictures in the training database, but once the training is done, the recognition task is performed relatively fast. The recognition process only depends on the neural network structure and not on the number of trained faces.

Much of the present literature on face recognition with neural networks presents results with only a small number of classes. Good results are reported, but the database is often quite simple, the pictures are manually aligned and there is no lighting variation, rotation or tilting. Hjelmås and Wroldsen describe a face recognition system using PCA for dimensionality reduction and feature extraction, and using a MLP neural network for classification [69]. They report of a correct classification of about 90% when using a test set containing 200 face images.

<b>Advantages</b>	<b>Drawbacks</b>
<ul style="list-style-type: none"> <li>+ Stores neighbourhood relationships</li> <li>+ Robust against noise and occlusion</li> <li>+ Robust against scaling, orientation and translation when face is correctly normalized</li> <li>+ Fast recognition/Low computational cost (depending only on the network and not the number of images)</li> </ul>	<ul style="list-style-type: none"> <li>- Sensitive to faulty normalization</li> <li>- Sensitive to illumination and face expressions</li> <li>- Sensitive to perspective, viewing angle and head rotation (can be improved using ensembles of networks)</li> <li>- Can be slow and difficult to train (especially for large databases)</li> </ul>

## 5 Hybrid Systems

This chapter will give a description to some of the hybrid approaches that have been investigated by different computer vision communities around the world.

### 5.1 Introduction

Hybrid approaches have a special status among face recognition systems as they combine different recognition approaches in an either serial or parallel order to overcome the shortcomings of the individual components. Different recognition approaches succeed and fail at widely different viewing and illumination conditions. Due to this dilemma it seems obvious to run various individual recognition classifiers on a problem leading to an individual ranking of the results of every process and to design a classification scheme to assess an overall recognition result. Another interesting approach in contrast to this parallel approach is a serial one where output from one classifier is input to the next. The serial approach can even go along with hybrid learning if the classifiers require training [106].

### 5.2 Hybrid Parallel Approaches

The idea of combining multiple inputs to infer information about the actual environment is a very natural method because it is done by humans every day. We combine acoustic, visual, tactile and thermal information to react on the world around us. Sometimes it is not even possible to derive the information wanted from one single sensor, but only in a joint effort it can be done. Because there is no perfect sensor it is reasonable to make use of the favourable properties of the individual sensors and suppress the disadvantages by applying a smart combination scheme (Figure 28).

#### 5.2.1 Combining Frontal and Profile Templates

Gordon described a hybrid system combining template-based frontal and profile face recognition in 1995 [107]. The approach extracts facial features to perform normalization and define template regions used for combined recognition of frontal and profile regions in a classical template matching process (Figure 26). After identifying head bounds in the frontal view, eye candidates are extracted using general eye templates, pupil detection and structural knowledge about the human head. A similar approach is used in the profile case by first extracting the profile line and then estimating the nose and chin tip. Overall template matching is subject to a scoring of five facial templates (left eye, right eye, nose, mouth and profile).

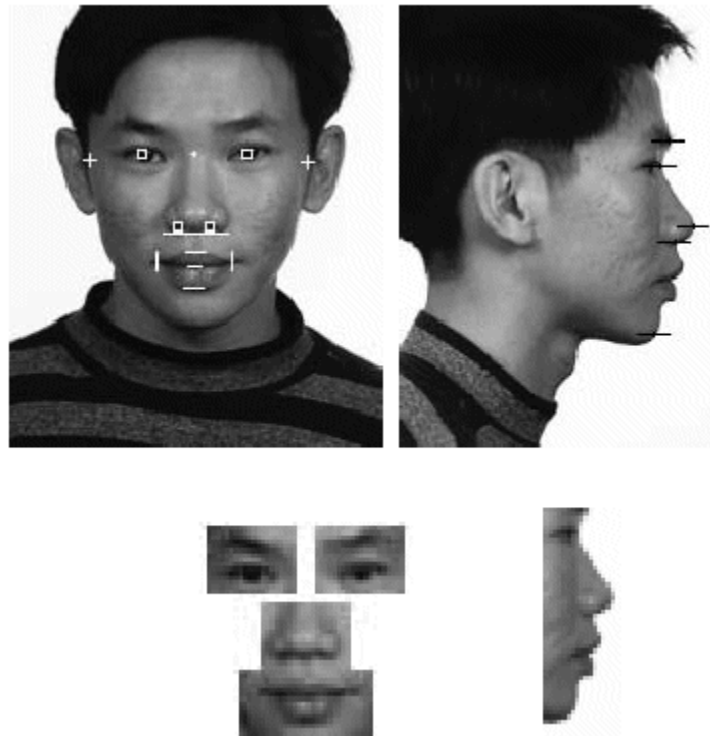


Figure 26: Frontal and profile templates for hybrid face recognition

### 5.2.2 Combining LDA and PCA

Marcalis and Roli at University of Cagliari recently described a face recognition system which was based on fusion of the two well know statistical methods LDA (fisherfaces) and PCA (eigenfaces) [108]. The first step consists of representing the face according to PCA and LDA in a regular manner (Figure 27). The distance vectors for PCA and LDA are computed for all the faces in the database. A final decision is made by combining these two vectors. Two algorithms, K-Nearest Neighbours (KNN) and Nearest Mean (NM), were proposed for the fusion phase. Reported results confirm the benefits of fusing PCA and LDA together. In general the performance of the KNN fusion rule was better than NM. It was also stated that more investigation was needed to understand the behaviour of combining PCA and LDA properly.

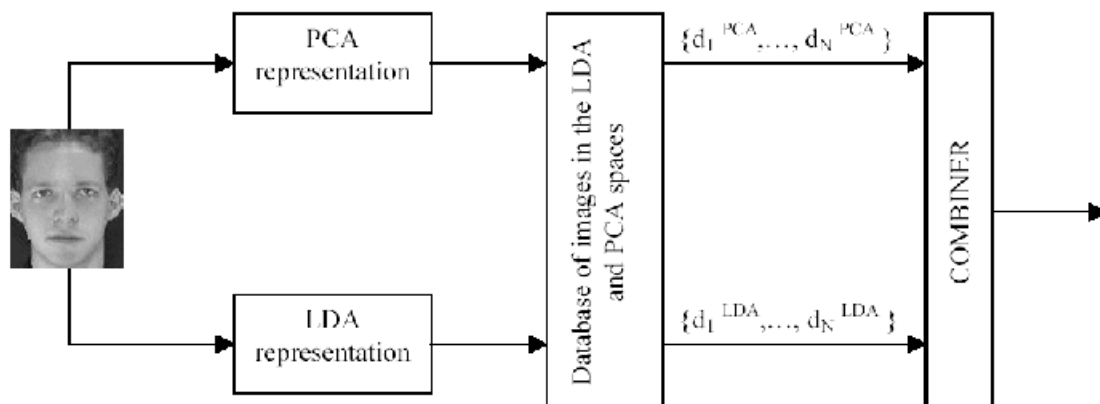


Figure 27: Hybrid fusion of PCA and LDA

### 5.2.3 Combining HMM, PCA and Profile Templates

A proper parallel hybrid face recognition system integrating three face classifiers was presented by Achermann and Bunke in 1996 [109]. Two full classifiers (HMM and PCA) and a profile classifier doing shape comparison was used. The recognition system thereby combined information from completely different information sources. Their tests showed that the performance of a combined classifier exceeded those of single classifiers, and for the test images a combination of three classifiers was superior to two classifiers.

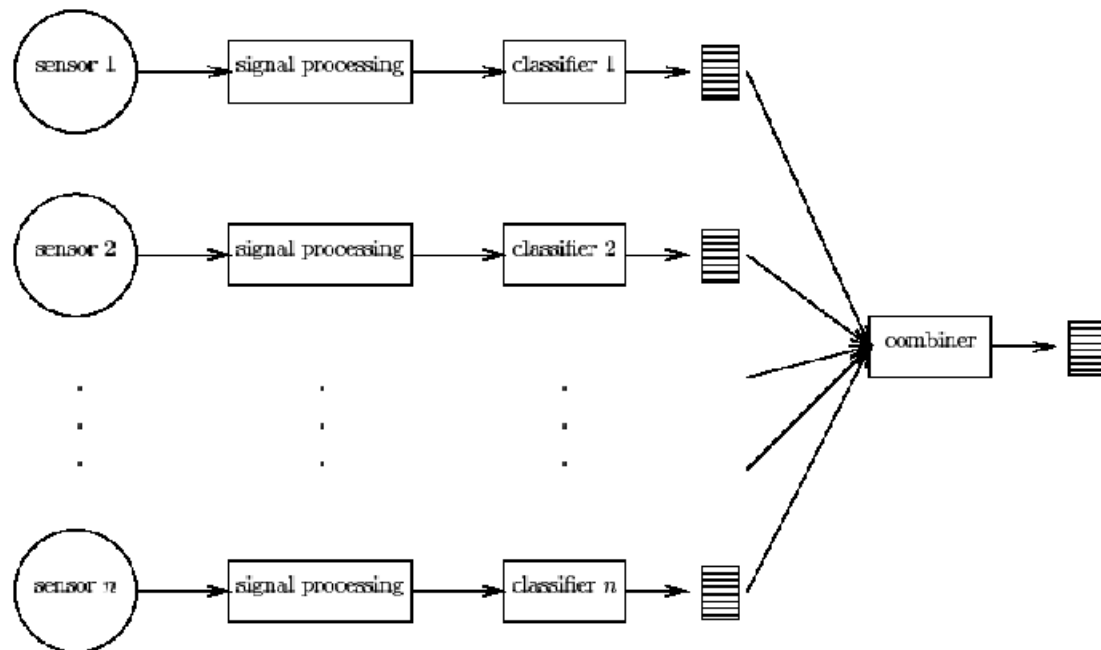


Figure 28: Hybrid parallel combination scheme

For combination schemes based on score functions it is necessary to make transformations of score values from the different classifiers in order to make them comparable. The scores can be distance measures, probabilities, quality measures etc. Ideally there should be a large gap between the score value of the best and second best class match, which indicates high evidence for correct identification. Some of the most common strategies for combining these scores are based on voting, ranking and scoring. With voting every classifier has one vote and a decision is reached through the majority of votes. If the voting ends in a draw the combination classifier is unable to decide for a certain class and a reject is returned. Voting methods only take into account a small part of the classifiers result since they are exclusively based on the first rank. Often there is a very high probability to end up in a draw. To avoid the disadvantages of transformations and use more information than only the top decision of a classifier, one can rely on ranking of the individual classifiers. The simplest approach is to compute the sum of the rank for every class in the combination set and the class with the lowest rank sum will be the best match. With scoring, information of the score function of an individual classifier is used to assess the ranking of the combination classifier. Scoring is favourable because it provides information not available through voting or ranking.

### 5.2.4 Combining with Other Biometrics

One area of research that will not be discussed any further in this report is the fusion of face image data with other types of biometric information. It is worth noticing that this may be the future solution for making a biometric system that is 100 % accurate. Human identification systems that are 100 % accurate will be needed to avoid fraud in critical security applications like ATM machines. Sanderson has written a report on fusion of speech and face information [110].

An approach close to studying visible images of the face is studying infrared images of the face. Wolff, Socolinsky, Eveland, Selinger and Neuheisel have found improved recognition performance using thermal images especially when having different illumination conditions [111][112][113]. While conventional cameras sense reflected light, thermal infrared cameras measure emitted radiation above room temperature. Thermal face scans are also robust to makeup and disguises. A face recognition system incorporating simultaneously registered thermal and visible images was developed and tested with good performance. The low interest in thermal imagery has been because thermal sensors are expensive, have lower resolution and produce more noise.

## 5.3 Hybrid Serial Approach

The serial approach on the other hand uses the output from one classifier as input to the next. The idea is to combine feature extraction or classifier algorithms which are complementary to each others weaknesses. The recognition process can also be implemented as a refinement process where the first classifier points out the most promising candidates, and the second classifier makes a more thorough investigation among those candidates to get a final result.

### 5.3.1 ERBF and Decision Trees

An example of a proper serial hybrid classification scheme involving hybrid learning was introduced by Gutta, Huang, Imam, Takacs and Wechsler in 1996 [87][104][114]. The presented system for face and hand gesture recognition combines ensembles of radial basis functions (ERBF) with decision trees (DT). Two different ERBF architectures were tested (Figure 29). ERBF1 separately trains the same three RBF nodes on three different sets of images (the original, the original distorted with Gaussian noise and the original distorted by geometric rotation).

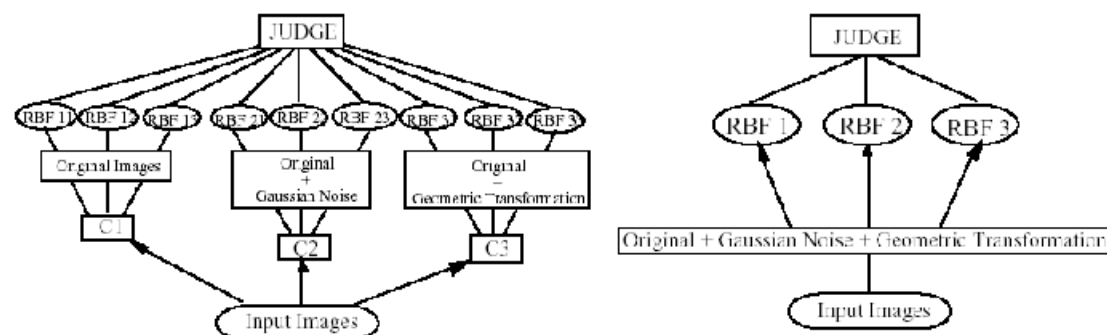


Figure 29: ERBF1 (left) and ERBF2 (right)

ERBF2 trains the same three RBF nodes on a combination of the imagery of ERBF1. The output from both networks consists of objects described by a fixed set of attributes and discrete values. The symbolic stage uses decision trees to derive rules for classifying these objects based on a collection of training objects with known class labels. The hybrid character in this approach involves that the training inputs for the DT method are generated from the already trained ERBF method. Results showed that ERBF outperformed simple RBF approaches, that hybrid learning improves classification performance and that training on a combination of original and distorted data (ERBF2) leads to improved performance against training on separate sets of training data (ERBF1).

### 5.3.2 Neural Network and HMM

Wallhoff and Rigoll experimented on recognizing profile views with a system trained on frontal views [115][116]. The recognition system combines a neural network and a hidden Markov model (Figure 30). A neural network is trained to output an artificial profile when a frontal image is set as input to the network. The classification process is based on the 1D hidden Markov model approach. One of the main ideas of the system is to do recognition without using any 3D information of heads and faces. It should be noted that recognizing a profile view by just having a frontal image is a real challenge even for humans. The system was tested on 100 people in the mugshot database with a recognition rate of 60%. To improve recognition rates larger training sets to the neural network are needed, but in this case recognition will never be very good because of the missing information.

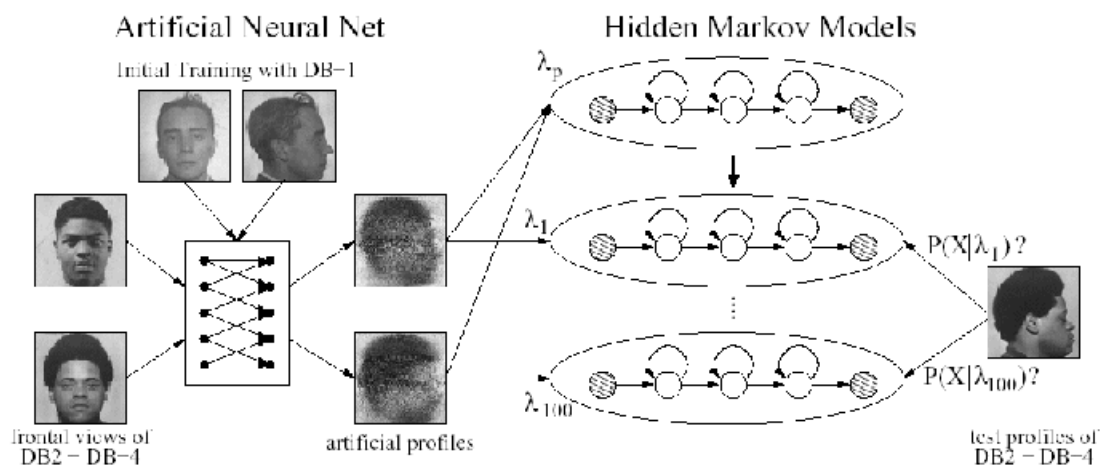


Figure 30: Hybrid serial approach with a neural network and HMM



## 6 Implementation

This chapter contains all information about my implementation design of a hybrid face recognition system. First a coarse outline of the system is described along with the used methods and motivations for the choices of these. Also a brief description of the tools we have used will be presented. Later a more detailed description of all the system modules and their functionality is given.

### 6.1 Choice of Hybrid Combination

From the research material studied in this report one can see that there has not been much focus on hybrid combinations for face recognition in the past. This is now changing and researchers are starting to believe that a hybrid combination of different recognition techniques is the only way to overcome the many difficulties encountered using single method face recognizers. Biological relevance is an important argument for why hybrid systems should be investigated further. Humans show a remarkable cleverness using hybrid combinations not only for face recognition, but also many other difficult tasks.

I decided to go for a serial combination of methods making up a hybrid face recognizer instead of the parallel combination. One of the main reasons of this choice is that the parallel approach is very time and resource consuming because of the way it is running different face classifiers in parallel. The serial approach is much more interesting and flexible because it passes information from one classifier to the next.

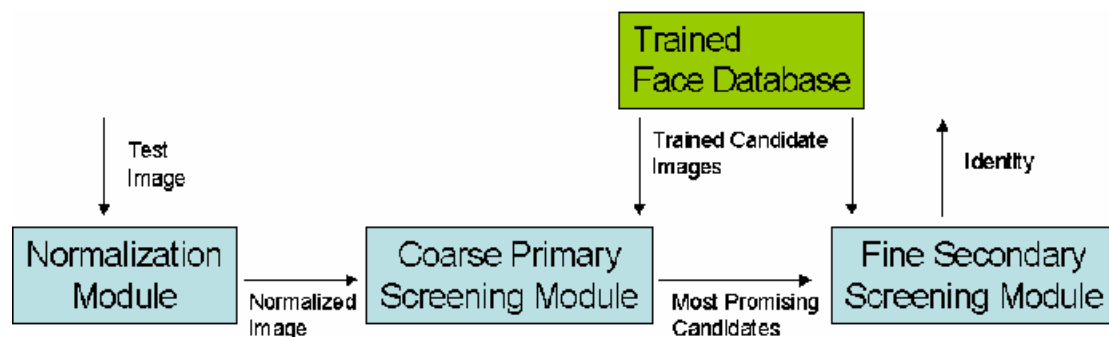


Figure 31: Intuitive new way of looking on the problem of hybrid face recognition

The implemented face recognition system consists of three main modules connected together in a serial manner (Figure 31). These modules are the normalization module, the coarse primary screening module and the fine secondary screening module. The division of face recognition into coarse and fine sorting the candidate images in the database introduces a completely new and intuitive way of looking on the problem of hybrid face recognition.

### 6.1.1 Normalization Module

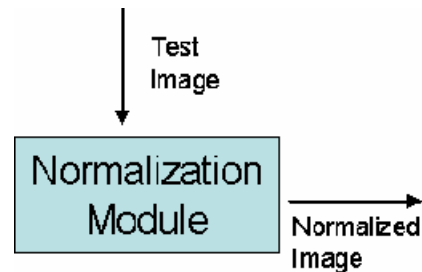


Figure 32: Flow chart of the normalization module

The first main module is the normalization module (Figure 32). It makes sure that all images that are used for face recognition enter the hybrid system in a uniform manner. This means that whenever a face image is added to the database by training or matched to the database during recognition it has to be normalized first. The most important property of the normalization module is the management of scale variations and vertical head rotations. Input images are scaled, cropped and rotated such that image sizes entering the system are exactly the same and they are therefore very easily comparable to each other. Another important feature of normalization is gained when cropping the image. This is a kind of information reduction. Removing the background and keeping only the faces or the most important parts of the faces reduces the amount of information needed to be processed further and removes information not useful for recognition. This helps speed up recognition times and increases recognition performance. Accurate and consistent normalization is maybe the most important step in creating a good face recognition system. Normalization of horizontal head rotations where faces have different views has been investigated and will be discussed later.

### 6.1.2 Coarse Primary Screening Module

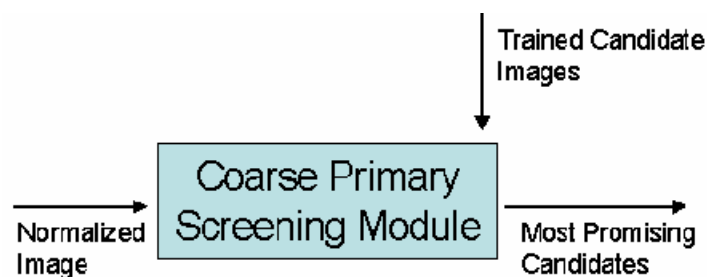


Figure 33: Flow chart of the coarse primary screening module

The second main module is the coarse primary screening module (Figure 33). This module receives a normalized image from the normalization module. During recognition its main task is to compare this input image with all the images stored in the database and reduce the amount of possible candidates. A large database can have extremely many stored images, typically thousands, quickly resulting in an overwhelming amount of processing need to be done. Comparison of a test image

against each of the trained candidate images must therefore be very quick, but without losing too much accuracy. All the images in the database get a score based on their similarity to the test image and the most promising candidates with the highest scores are chosen for further processing. It is important that the candidate screening process ensures with very high confidence that the correct identity is among the promising candidates and is not filtered out. Ideally this confidence should be 100%.

Based on the earlier evaluation of the different techniques a choice of appropriate method for this module was made. PCA based eigenfaces and LDA based fisherfaces run into some problems when the database gets too large. Not all images can participate in creating the eigenvectors and eigenvalues needed to span up eigenspace. The main reason behind this is that the covariance matrix gets too large. One trick will be to use only a selected number of training images for creating eigenspace. This will often result in that images not used will have features that may not be handled very well by the modes of variation and recognition performance will drop. Another reason for not choosing these methods are the difficulties of adding completely new images to the database of trained images. The same problem occurs as before, because new features not trained before are bad represented in eigenspace. Retraining eigenspace is a very costly process and should be avoided unless it is really necessary.

Neural networks are biologically inspired by human neurons, and are especially clever at handling large datasets. An enlarged database implies that the size of the neural network can remain the same and only the weights between the nodes are changed. This is one of the main advantages of using a neural network together with a large database. The speed of recognizing a test image using a neural network is also superior to other techniques. A neural network was not chosen because it delivers only one believed output and gives no scoring information that can be used for selecting the best candidates. Another important disadvantage is the problem of adding new pictures and identities to the database. The whole network and its weights need to be retrained if it should be able to recognize the new individual. Retraining a neural network is a very time demanding process.

Both HMM and Gabor wavelets provide methods for incremental learning of new classes. This means that new faces can be added to the database without recomputing the representations of all other learned faces. A trained face image database having the incremental learning feature is extremely desirable for large databases. When comparing computational load matching an image to the trained database HMM is the most time consuming one. The computational complexity when using Gabor wavelets is much easier to adjust by for instance changing the number of jets used for image comparison. It was decided to implement the coarse primary screening module using Gabor wavelets because it can be very quick and delivers a separate similarity score for each compared image in the training database. Gabor wavelets are good at handling different illumination conditions and also at handling noisy pictures. They are not so good at identifying people when they have different facial expressions or when some parts of the face is occluded.

### 6.1.3 Fine Secondary Screening Module

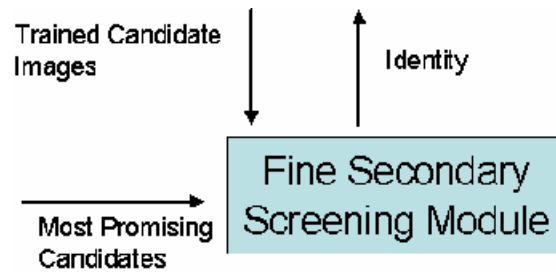


Figure 34: Flow chart of fine secondary screening module

The third main module is the fine secondary screening module (Figure 34). This module receives the names of the most promising candidates, the ones with the highest similarity scores, from the coarse primary screening module. During recognition its main task is to find out the correct identity of the individual on the test image presented. The number of candidates to be considered has now been reduced significantly by the coarse secondary screening module, which means that comparison of each remaining image is allowed longer processing time. The focus of this module is more on recognition accuracy than on recognition speed. All earlier problems concerning handling large databases are now completely gone because of the limited set of promising candidate images that need further investigation. Specific trained candidate images are requested from the face image database when needed.

The technique used for fine sorting the images should be chosen so that it in a best possible way overlaps and complements with the Gabor wavelet technique used in coarse sorting. A technique that is good at handling facial expressions and occlusion would therefore be a preferable one.

PCA based eigenfaces is a robust method handling both occlusions and facial expressions. It is also very good at handling small datasets. The earlier discussed problem concerning exclusion of some training images can now be ignored because of the limited training set. All trained candidate images are being used when creating eigenspace and this results in that all faces and their features can get a best possible linear combination representation. These were some very good reasons for choosing to implement the fine secondary screening module using eigenfaces. An important new idea is spanning up eigenspace during recognition using the candidate images, and not before execution using the whole database as usual. This can be done only because of the limited dataset of candidate images, which reduces training time dramatically.

LDA based fisherfaces is a robust method handling both occlusions and facial expressions. The reason why it was not chosen is because the fisherfaces method uses particular class information where it is recommended to have many images per class in the training process. This will not be the case in the fine sorting process where only a limited number of candidate images are left for processing. When only a few images are used combined with LDA the result is lousy recognition performance.

Neural networks are also quite good at handling occlusions, but show more sensitivity to facial expressions. This is one reason for not choosing neural networks. Another more important reason deals with intricate implementation issues. There is a common misapprehension that if the algorithms and data structures get complex enough then a complex problem will get solved. This is almost never right. I believe that the focus must be on simple but powerful algorithms which are easy understandable.

HMM has shown better recognition performance than eigenfaces when tested with the same training set. Facial expressions are handled satisfactory, but the method shows sensitivity to occlusions. To estimate a good model for recognition as much training data as possible is needed in the estimation of hidden Markov model parameters. A good initial model is important when it comes to making correct recognition decisions. This may be more difficult with a limited dataset. The obvious solution is to have the trained database of Gabor jet images also include HMM trained models of people. Before recognition can take place both Gabor jet images and HMM models have to be trained in advance. The complexity of such a system makes the final choice of method easier. I decided to go further with the eigenfaces method because it is very understandable, less complex and does not require any database addons.

## **6.2 Software Engineering Tools**

This subchapter contains a simple description of the tools that were used. The implementation has been done mostly using Matlab version 6.5 and the accompanying Image Processing Toolbox version 4.

### **6.2.1 Matlab 6.5**

Matlab is a simulation environment for doing numerical computations with matrices and vectors (Figure 35). It handles a wide range of computing tasks in engineering and science, and has several built-in interfaces that let you quickly access and manipulate data. The Matlab environment integrates mathematical computing, visualization and a powerful technical language. A large user community spread throughout industry, government and academia makes Matlab a recognized standard worldwide for technical computing. Matlab is used in a variety of application areas including signal and image processing, control system design, earth and life sciences, finance and economics and instrumentation. The open architecture makes it easy to use Matlab to explore data and create custom tools. One interesting feature is that Matlab applications can be converted to standalone applications using the C and C++ compiler.

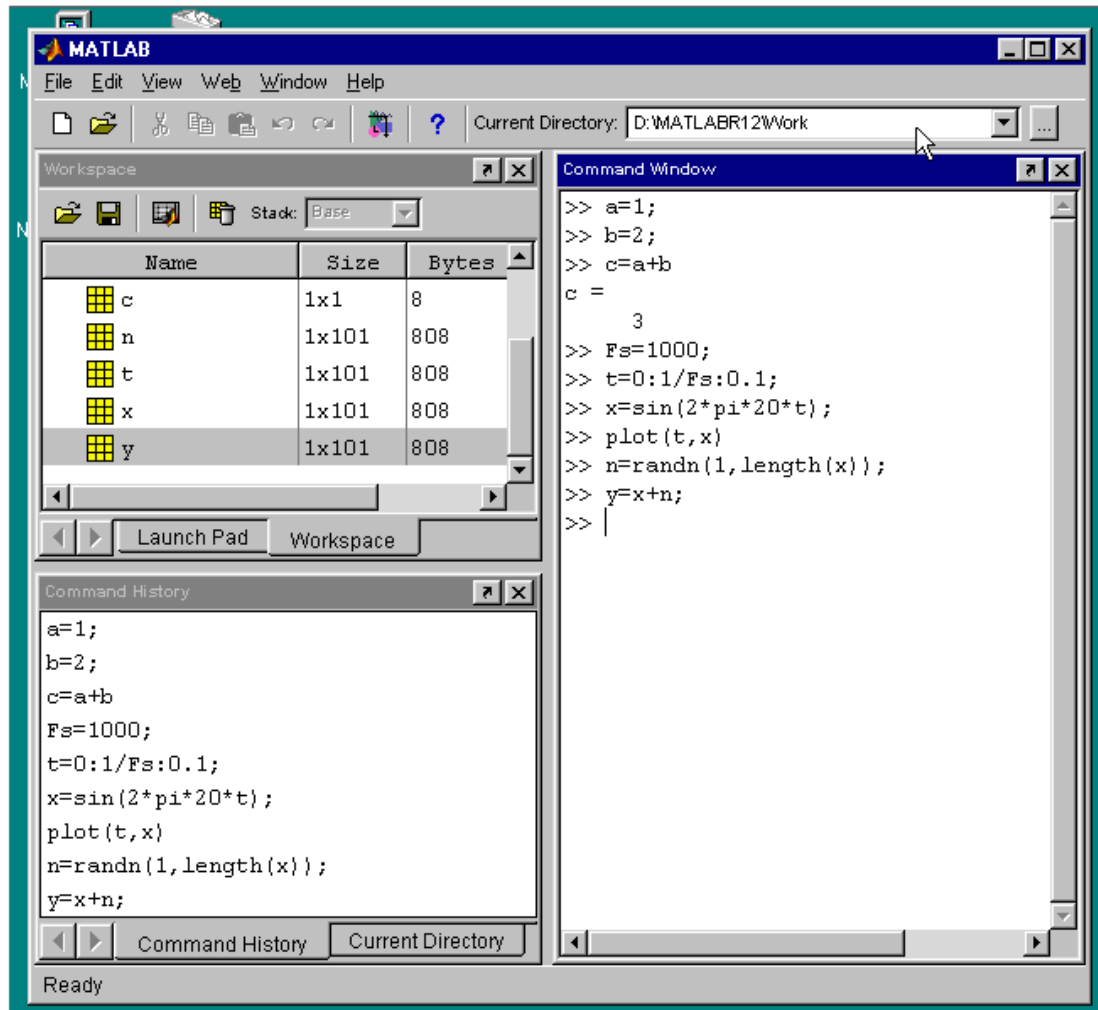


Figure 35: Matlab graphical user interface and running a simple program

## 6.2.2 Image Processing Toolbox 4

The image processing toolbox extends the Matlab computing environment to provide functions and interactive tools for enhancing and analyzing digital images and developing image processing algorithms. Most functions are implemented in the open Matlab language, which lets you explore and customize existing toolbox algorithms or you can develop your own algorithms.

## 6.3 System Modules

This subchapter contains a detailed description of all the implemented system modules and their individual functionalities.

### 6.3.1 Face Normalization (facenorm.m)

The facenorm module takes a number as input. The number relates to a specific face image that needs normalization. First some important normalization parameters are defined.

Global normalization definitions explained:

<b>dataPath = .\DATA\</b>	Path to data files like COORDS3816.TXT and M150X130.DAT
<b>coordFile = COORDS3816.TXT</b>	Name of the eye coordinate file containing coordinates for 3816 face images.
<b>maskFile = M150X130.DAT</b>	Name of the mask file to be convoluted with the image during normalization.
<b>imagePath = .\FERET-PGM\</b>	Path to original images
<b>imageFormat = .pgm</b>	File extension of original images
<b>normPath = .\NORM\</b>	Path to normalized images
<b>normFormat = .nrm.pgm</b>	File extension of normalized images
<b>eyedistance = 70</b>	Distance between the eyes after normalization
<b>eyerow = 45</b>	Vertical distance to a line between the two eyes
<b>norm_height = 150</b>	Image height after normalization
<b>norm_width = 130</b>	Image width after normalization

All the information about file names and corresponding eye coordinates are collected from the predefined coordinate file COORDS3816.TXT and placed into proper arrays. Then the selected image number  $n$ , which now relates to a specific file name, is read from the original images directory (\*.pgm). The first step in the normalization of the incoming image will be to horizontally align the right and left eye (Figure 36). This is done by rotating the image an angle  $\alpha$ . Alpha was defined as  $\arctan(\text{hdiff}/\text{wdiff})$ , where  $\text{hdiff}$  is the vertical eye coordinates difference and  $\text{wdiff}$  is the horizontal eye coordinates difference. A positive angle will result in a counter clockwise rotation and a negative angle will result in a clockwise rotation of the image.

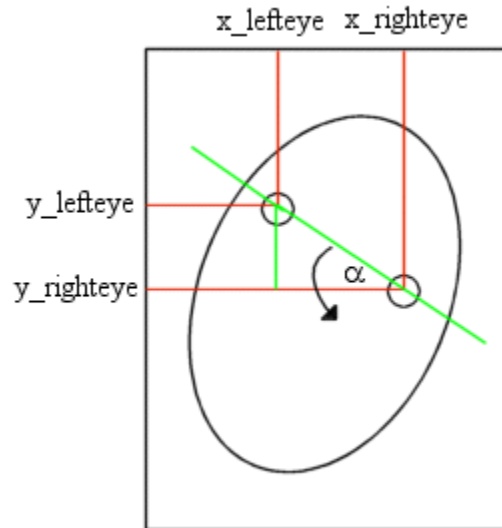


Figure 36: Horizontal alignment of the right and left eye by rotating the image

The next step is resizing the image based on the distance between the left and the right eye. To be able to calculate the scale factor first we have to find out the new eye coordinates due to the previous rotation (Figure 37).

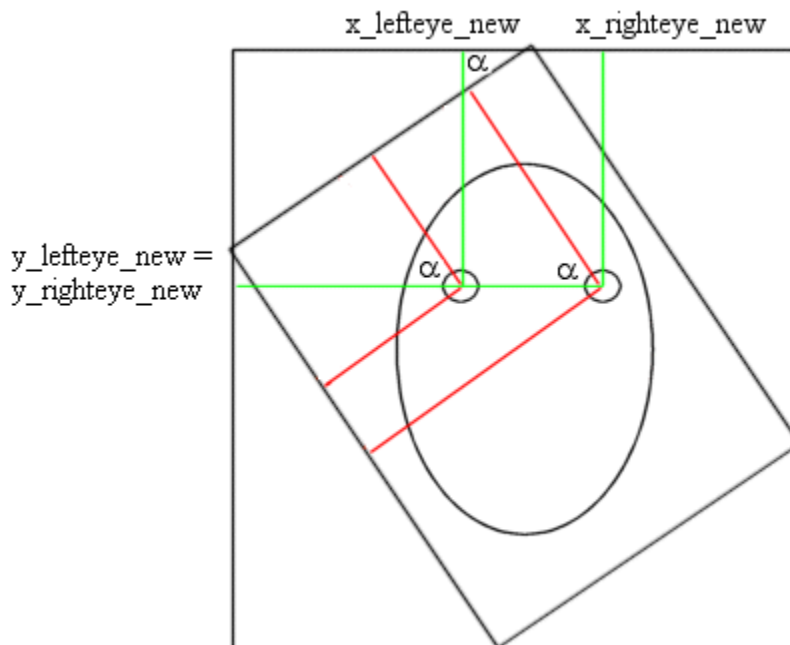


Figure 37: Finding the new eye coordinates in the rotated image

**New eye coordinates for counter clockwise positive rotation  $\alpha$  :**

$$x_{\text{new}} = x * \cos(\alpha) + y * \sin(\alpha)$$

$$y_{\text{new}} = (\text{imagewidth} - x) * \sin(\alpha) + y * \cos(\alpha)$$

**New eye coordinates for clockwise negative rotation  $\alpha$  :**

$$x_{\text{new}} = x * \cos(-\alpha) + (\text{imageheight} - y) * \sin(-\alpha)$$

$$y_{\text{new}} = x * \sin(-\alpha) + y * \cos(-\alpha)$$



After having found the new eye coordinates the image is then normalized with respect to scale by making the distance between the eyes equal to the defined eyedistance of 70 pixels. This is done by scaling the entire image a factor  $\text{eyedistance}/\text{wdiff}$ , where  $\text{wdiff}$  is the new horizontal eye coordinates difference.

The third step is cropping the image to the desired size defined by  $\text{norm\_height}$  and  $\text{norm\_width}$ . I have used 150x130 pixels which should correspond well with the eyedistance of 70 pixels. The parameter  $\text{eyerow}$  defines how many pixels in vertical direction above the eyes the cropping should start. 45 pixels is a good choice of  $\text{eyerow}$  when the eyedistance is chosen to be 70 pixels.

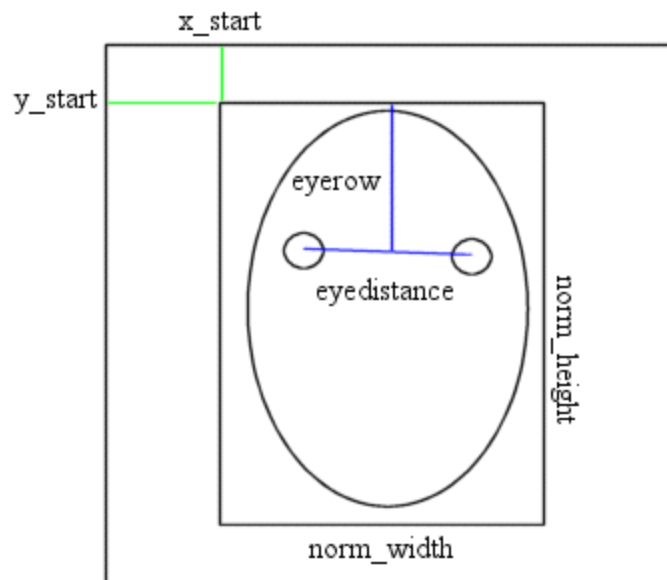


Figure 38: Cropping the image using  $\text{eyerow}$  and  $\text{eyedistance}$

#### Start coordinates used for cropping :

$$x_{\text{start}} = x_{\text{lefteye\_new}} * \text{scale} - (\text{normwidth} - \text{eyedistance})/2$$

$$y_{\text{start}} = y_{\text{lefteye\_new}} * \text{scale} - \text{eyerow}$$

The fourth step is convoluting the cropped image with a face mask defined by the parameter  $\text{maskfile}$ .  $\text{M150X130.DAT}$  convolution mask removes background information and saves only the pixel information from the oval face area.

The fifth step does a regular histogram equalization of the remaining pixels, which distributes the different greyscales in the histogram according to their occurrence in the picture. One effect is that parts of the image with more frequency variation will be more enhanced, while parts with less frequency will be neglected. Finally a copy of the normalized image is written to the normalized faces directory (\*.nrm.pgm).

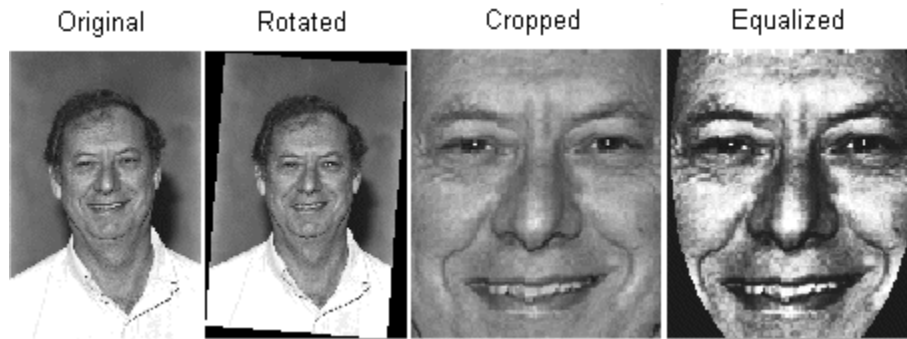


Figure 39: Example normalization of a face image

### 6.3.2 Batch Face Normalization (facenormall.m)

The facenormall module batch normalizes all the images referenced by the coordinate file COORDS3816.TXT sequentially. All the steps described in the above facenorm module are repeated n times, once for each of the individual face images referenced.

### 6.3.3 Top of Face Normalization (facenormtop.m)

The facenormtop module processes only the top of the head consisting of the forehead, eyes and nose (Figure 40). Mouth and chin which represents the most dynamic face features are efficiently excluded from the following recognition process by removing them as early as in the normalization process. The process is identical to the facenorm module, except the parameter norm\_height which is reduced from 150 to 100 pixels. A performance comparison of using the whole face or just the top part of the face is discussed later in the results chapter.



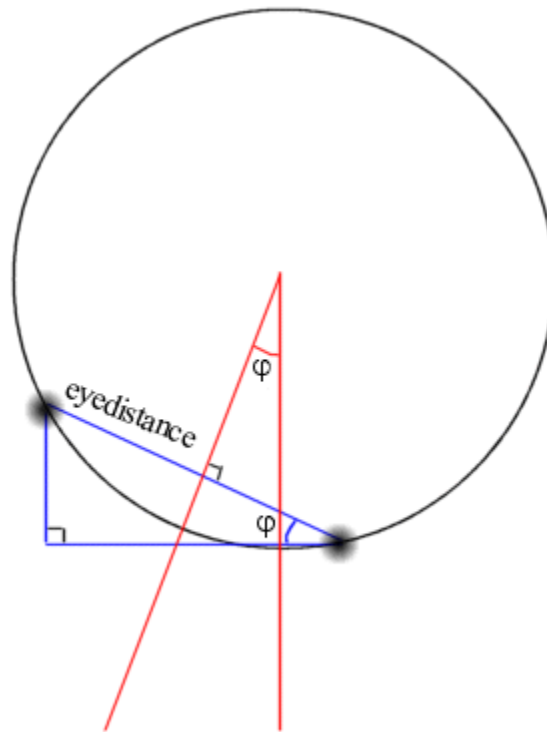
Figure 40: Example normalization of the top of the face

### 6.3.4 Batch Top of Face Normalization (facenormtopall.m)

The facenormtopall module batch normalizes all the images referenced by the coordinate file COORDS3816.TXT sequentially. Only the top of the head is normalized and saved for each referenced image. The normPath parameter is set to .\NORMTOP\ to avoid conflicts with the general face normalization module.

### 6.3.5 View-Based Face Normalization (facenormrot.m)

The normalization method described above is intended only for head on images, and has to be modified to handle face images taken from different views. One problem with normalizing rotated face image arises when trying to scale to achieve a fixed distance between the eyes. A fixed eyedistance of 70 pixels will obviously be wrong for different poses and a good solution is therefore to use a specific distance calculated for each pose separately. The goal of the scale normalization process is to produce images where the face of one person has the same size in all poses.



**Figure 41: Rotated face seen from above.**

Handling rotated face images can be studied in general (Figure 41). The vertical line on the figure points in the direction of the viewer and the sloping line points in the direction of the nose. The two dark dots indicate the positions of the left and right eye. An image taken of a rotated face will capture the distance between the eyes as  $\cos(\varphi)$  times the actual eyedistance, where  $\varphi$  is the head rotation angle. This assumption will only hold if the image is parallel projected on a surface perpendicular to the viewing direction. In reality this will never be true for a photograph, which is more likely a perspective projection, but it will be a reasonable approximation if the camera is sufficiently far away from the object.

The facenormrot module has been specifically designed to work together with the FERET image database, which contains some series of rotated face images. Rotated image series are labelled ba (0), bb (+60), bc (+40), bd (+25), be (+15), bf (-15), bg (-25), bh (-40) and bi (-60). The numbers inside the brackets following the labels indicate an approximate rotation angle in degrees from which the images have been captured. Positive rotations indicate images of subjects which face left (photographers right), while negative rotations indicate images of subjects which face right

(photographers left). The distance from the subject to the camera however is not mentioned in any FERET documentation.

Experiments show that using the approximate angles defined will lead to faulty normalization, and for that reason they have not been used. Instead I made the assumption that each series of face images from the same individual has been captured with the approximate same distance between subject and camera. This implies that the head rotation angle  $\varphi$  can be inferred on the basis of the eye distance for a head on image (ba) compared to the eye distance for a rotated image (bb, bc, bd, be, bf, bg, bh, bi). Each of the rotated images will then get an associated head rotation angle  $\varphi = \arccos(D_{ba}/D_{bx})$ , where  $D_{ba}$  is the eye distance for the head on image (ba) and  $D_{bx}$  is the eye distance for the rotated image (bx).

The scale used for normalization is the same for all rotations in each of the series, and is based entirely on the scale calculated for the head on image (ba). The cropping process is more intricate and depends on the calculated head rotation angle  $\varphi$ . We are interested in creating a smooth transition in normalization when rotating the head. At the same time we want to keep the important parts of the face, containing eyes, nose and mouth, inside the cropped area. The term  $20\tan(\varphi)$  in the equations below makes sure that a  $\varphi$  degree head rotation results in a horizontal cropping displacement. For a 45 degree head rotation the displacement will be exactly 20 pixels, and if there is no head rotation this term will nicely cancel itself out.

#### **Start coordinates used for cropping :**

$$\varphi=0 : x_{start} = x_{lefteye\_new} * scale - (normwidth - eyedistance)/2$$

$$\varphi>0 : x_{start} = x_{righteye\_new} * scale - (normwidth - eyedistance)/2 - normwidth - 20\tan(\varphi)$$

$$\varphi<0 : x_{start} = x_{lefteye\_new} * scale - (normwidth - eyedistance)/2 + 20\tan(\varphi)$$

$$\forall\varphi : y_{start} = y_{lefteye\_new} * scale - eyerow$$

The input to the facenormrot module is a number corresponding to one of the image series, containing 9 separate images with different head rotations. Eye coordinates for rotated images are found in the ROTATED.TXT file in the .\DATA\ directory. The image is normalized as described and the resulting normalized images are stored in the .\NORMROT\ directory. The results of the normalization will also be presented to the user in form of a plot of all the images normalized (Figure 42). To normalize all the images referenced by ROTATED.TXT simply input the number 0.

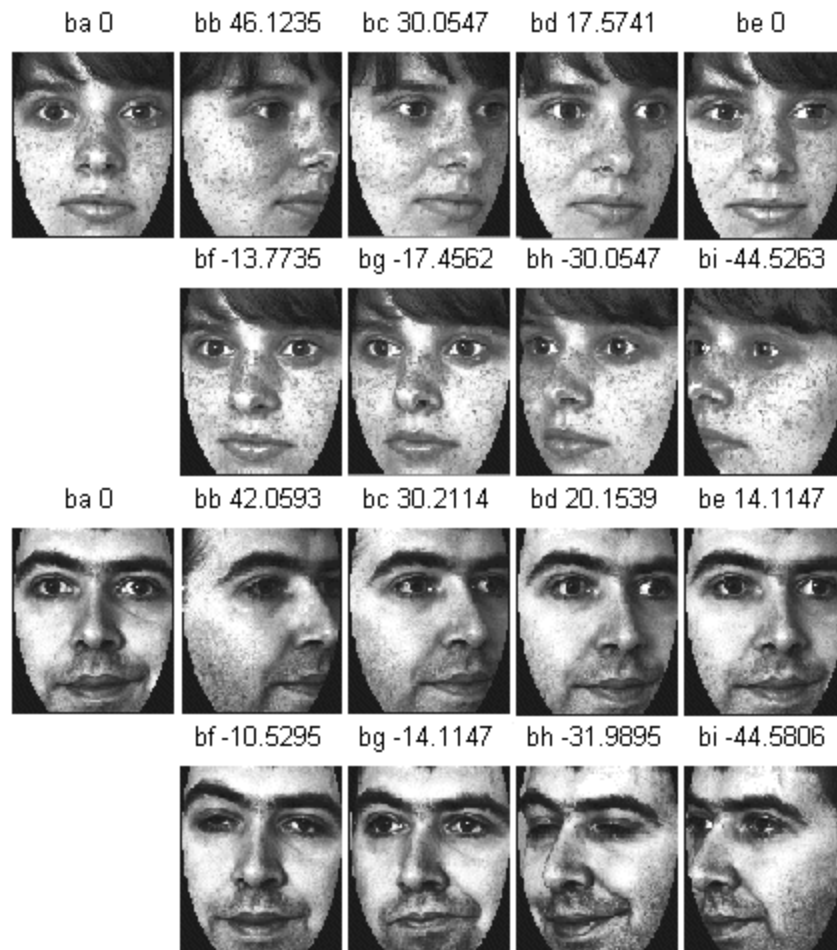
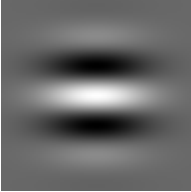
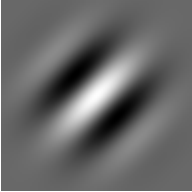
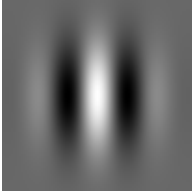
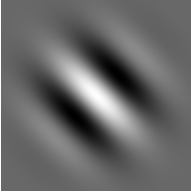
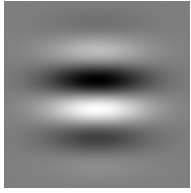
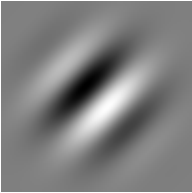
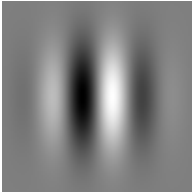
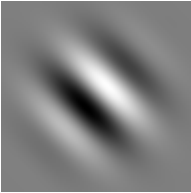


Figure 42: Two normalized head rotated image series and their corresponding rotation angles.

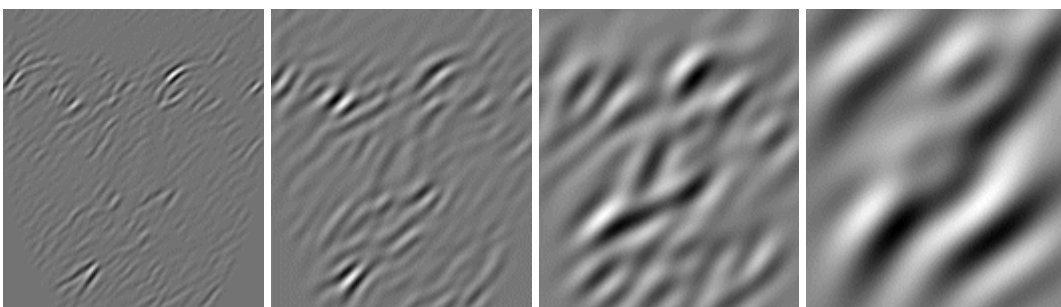
### 6.3.6 Gabor Training (gabortrain.m)

The gabortrain module trains the previously normalized images by extracting a set of Gabor jets from each of the images and storing them in a database. Gabor wavelet mask files of four different sizes and four different rotations were used, as well as both even and odd masks (Table 1). This means that a total of 32 masks were used. The mask rotation angles are 0, 45, 90 and 135 degrees, but can be increased to eight or more if necessary. The mask sizes are 11x11, 23x23, 47x47 and 95x95 pixels, and can also be chosen as preferred. Cosine (even masks) and sine (odd masks) with same rotation and scale have opposite phases, thereby complementing each others actions. One should remember that increasing the number of rotations and/or sizes also increases computational complexity and thereby recognition time. A set containing 32 masks seems like a reasonable minimum showing good performance. The even mask files are implemented in apply\_mask0even.m - apply\_mask15even.m and the odd mask files in apply\_mask0odd.m - apply\_mask15odd.m.

	0 degrees	45 degrees	90 degrees	135 degrees
<b>Cosine Wave</b> [Even Mask]				
<b>Sine Wave</b> [Odd Mask]				
<b>95x95 pixels</b>	apply_mask12	apply_mask13	apply_mask14	apply_mask15
<b>47x47 pixels</b>	apply_mask8	apply_mask9	apply_mask10	apply_mask11
<b>23x23 pixels</b>	apply_mask4	apply_mask5	apply_mask6	apply_mask7
<b>11x11 pixels</b>	apply_mask0	apply_mask1	apply_mask2	apply_mask3

**Table 1: The 32 Gabor wavelet masks and their mask numbers**

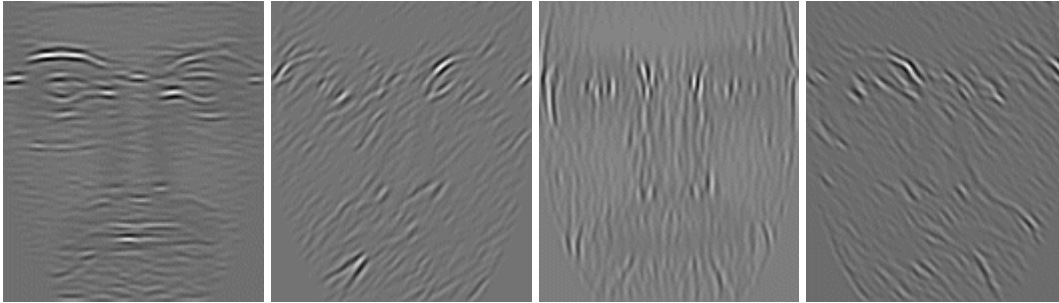
To really understand what is going on we first have to look at regular Gabor wavelet convolution where every pixel in the original image is convoluted with one predefined mask file. The example below shows convolutions with a 45 degree mask (Figure 43). All face features which lie in an approximately 45 degree angle will be emphasized and other angles will be de-emphasized. The different mask sizes will emphasize face features which match in size and de-emphasize face features that are either much smaller or much larger than the mask size. In convolution with mask1 we can see the small face features like eyes and mouth, while only the large face features have any influence on the result when convoluting with mask13.



**Figure 43: Convolution with 45 degree even masks with different mask sizes**  
(a) mask1 11x11 pixels (b) mask5 23x23 pixels (c) mask9 47x47 pixels (d) mask13 95x95 pixels

In a second example we show what happens if we try to change the rotation of the convolution mask (Figure 44). The mask size is kept constant at 11x11 pixels. We can observe that face features which match the direction of the convolution mask are emphasized. A good illustration of this is convolution with mask0 which clearly

emphasizes the horizontal shape of the mouth. When convoluting the normalized image with mask2 the opposite effect is observed when the mouth is de-emphasized.



**Figure 44: Convolution with even masks of size 11x11 pixels with different rotations (a) mask0 0 degrees (b) mask1 45 degrees (c) mask2 90 degrees (d) mask3 135 degrees**

2D Gabor functions are thus quite similar to enhancing edge contours, as well as valleys and ridge contours of an image. The most important face features like eyes, mouth and nose edges are enhanced, as well as moles, dimples and scars. An image can be represented by Gabor wavelet responses by convolving Gabor filters of different spatial frequency/size and orientation/rotation.

A full image convolution will create a dataset that is 32 times larger than the original image and is thus intractable for recognition purpose. The solution is simply to sample only the Gabor responses belonging to a few pixels in the image and not all of them. An evenly spaced grid was used for this purpose. This means that the stored Gabor jets are responses from applying the 32 different Gabor convolution masks to each of the pixels in the grid. Experiments made with different spacing between these feature points will be discussed in the results chapter. For a full head 10 pixel spacing equals  $13 * 15 = 150$  jets, 20 pixel spacing  $7 * 8 = 56$  jets and 40 pixel spacing  $4 * 4 = 16$  jets. When using only the top of the head 10 pixel spacing equals  $10 * 13 = 130$  jets, 20 pixel spacing equals  $5 * 7 = 35$  jets and 40 pixel spacing equals  $4 * 3 = 12$  jets. A set of Gabor responses coming from an image is called a Gabor jet image. Training these Gabor jet images in advance will reduce the computational cost in the recognition stage. The output data is stored in \*.mat files, which is a standard Matlab file format for storing variables to disk.

### 6.3.7 Gabor Recognition (gaborreq.m)

The gaborrec module is an implementation of the coarse sorting module described earlier. Input to the module is the parameter file which contains file names of all the images in stored in the database, and the parameter number which indicates which of them is used as a test image. This means that when recognizing an image exactly that image must be excluded from the training set, and thereby also from comparison. The incoming test image is first processed exactly like the trained images have been processed by the gabortrain module. All 32 masks are applied in an evenly spaced grid with the same spacing as the trained images to create a Gabor image representing the original. This Gabor image is then compared to each of the stored Gabor images in the trained database. When comparing the two images similarity measures for each compared jets are calculated. A combined similarity measure for the whole image is

found by simply dividing the sum of all similarity measures with the total number of jets in one image.

Different similarity measures for computing similarity between Gabor jets exist. Due to phase rotation, jets taken only a few pixels from each other have very different coefficients, although representing almost the same local feature. This can cause severe problems for matching. We can therefore either ignore phase or compensate for its variation explicitly. Using the phase has two potential advantages. Firstly, the phase information is required to discriminate between patterns with similar amplitudes, should they occur. Secondly, since phase varies so quickly with location it provides means for accurate jet localization in an image. We have used a similarity measure that ignores phase and uses only the magnitude information (Equation 1). We later will show some of our results when using this simple but still powerful similarity measure.

$$\text{Equation 1: } S_a(J, J') = \frac{\sum_j a_j a'_j}{\sqrt{\sum_j a_j^2 a'^2_j}}$$

The magnitude/amplitude  $a_j$  and the phase  $\phi_j$  is calculated on the basis of the even (cosine) and odd (sine) wavelet pair just as with complex numbers (Equation 2 and Equation 3). Our similarity measure needs the amplitude of 16 different combinations (4 rotations and 4 scales) for every jet.

$$\text{Equation 2: } a_j = \sqrt{\text{EvenWave}_j^2 + \text{OddWave}_j^2}$$

$$\text{Equation 3: } \phi_j = \arctan\left(\frac{\text{OddWave}_j}{\text{EvenWave}_j}\right)$$

The similarity measure for each image in the trained database is stored in a similarity vector which is sorted by descending similarity. Output from the gaborrec module is two sorted vectors, candval with the similarity measures and candname with the corresponding numbers referencing filenames.

### 6.3.8 PCA Training (eigenrain.m)

The eigenrain module receives only a subset of the candidates. I decided to process the hundred best candidates and to throw away the rest, but the number of candidates can be tuned freely based on the recognition performance and available processing time if necessary. A reduced subset of candidates removes all problems concerning large databases, and makes it possible to do a more thorough investigation of the remaining images. Training the promising candidates is done on the fly, which means that it is done during recognition.



Pixel values from all the images are first imported into a large trainingSet matrix where each column, which contains pixel values, belongs to a certain image. A trainingMean vector is created which contains pixel value means for all the different positions in the image. The trainingMean is then subtracted from the trainingSet to centre the pixel values. This means that the trainingSet matrix now describes how each pixel value differs from the mean pixel value for that exact position in the image. The covariance matrix, which is used to calculate eigenvectors and eigenvalues, equals the trainingSet matrix multiplied by its transpose. Eigenvectors are then sorted by descending eigenvalues, and the largest eigenvectors are used to span up eigenspace. For each image in the training set a linear combination is calculated and stored into the trainingBase matrix. The trainingBase matrix, the trainingMean vector and eigenspace A is thereafter temporarily stored to disk into facebase.mat in the .\DATA\ directory.

What we have achieved can be imagined as a 3D cloud of data, where each spot in the cloud is an image. The idea is that images from the same individual will form small clouds inside this large cloud, and hopefully will these small clouds not interfere too much with each other. Of course our images produce N-dimensional clouds. The first eigenvector, the one with the highest eigenvalue, will then point from the centre of the cloud in the direction where the discernability in the dataset of images is highest. This will be our first dimension. The second eigenvector, with the second highest eigenvalue, will point in the second best direction of discernability etc. Each image in the cloud can thereby be described by a linear combination of the eigenvectors.

### 6.3.9 PCA Recognition (eigenreq.m)

The eigenrec module makes the final decision about the identity of the person on the received test image. First a linear combination for the test image has to be calculated in the same manner as for the training images in the eigentrain module. The temporary trainingMean vector and eigenspace A is loaded from facebase.mat and used for this purpose. When this is completed we have got a representation of the unknown image which can be projected into the same N-dimensional space as has been done with the training images.

The test image is compared to all the images in the candidate cloud to find the one that has the closest match. Several different distance measures exist for this purpose. One is the standard L2-norm, also called Euclidean distance, which sums the squared pixel difference between training images stored in trainingBase and the test image stored in testBase (Equation 4). The combination which returns the smallest L2-norm value will reveal the best candidate. In terms of clouds in N-dimensional space this will be the shortest straight line connecting the test image point with its neighbour point in the local point cloud.

**Equation 4:** 
$$L2norm(m) = \sum_{i=1}^N (trainingBase(i,m) - testBase(i,1))^2$$

Another measure is the Mahalanobis distance which uses the eigenvectors as weighting for the contribution of each of the axis in eigenspace (Equation 5). The

final choice of classifier was a combined one, where L2norm is run if Mahalanobis fails. This is discussed later in the results chapter.

$$\text{Equation 5: } Mahalanobis(m) = -\sum_{i=1}^N (trainingBase(i,m) \cdot testBase(i,1) / \sqrt{eigenvalue(i)})$$

Finally the eigenrec module returns the name of the best candidate and its corresponding value, either Mahalanobis distance or L2norm.

### 6.3.10 Test Gabor Wavelet Recognizer (testgabor.m)

The testgabor module was made to test the primary coarse recognizer. It loops through all the images specified and performs a recognition decision for each of them using the Gabor recognizer. If you only want to recognize one picture simply remove the loop. The Gabor recognizer will return a sorted candidate list of names and values referencing all the images in the database. Since we have extra file name knowledge about what is really a correctly recognized image, we can find out the first position in the list having a correct match. This gives information about how good the coarse recognizer is, which can be used for tuning the module to produce even better results.

### 6.3.11 Test Hybrid Combination Recognizer (testhybrid.m)

The testhybrid module was made to test the performance of the full hybrid combination consisting of both primary coarse screening and secondary fine screening. The module loops through all the images in the database and makes a recognition decision for each of them, concerning their identity. The test images are read and processed with the gaborrec module as described in the previous section. In addition to this the hundred best candidates are selected and trained with the eigentrain module. Finally a definitive decision is made by the eigenrec module. The best match is compared with the real answer based on the file name and False or OK is printed to the screen together with other information about the recognition parameters.

## 7 Results

This chapter contains all information about the results coming from some different test made on the implemented design. The focus has been on trying to get an overview of how good my hybrid implementation is, and what can be done to improve its recognition performance. Especially I was interested in finding out why it went wrong in some cases.

### 7.1 Normalizing the Whole Face or Just the Top?

One important question is whether one should normalize the whole face or just the top part of the face. The most dynamic features in a face are the mouth, cheeks and eyes (Figure 45). These features can participate together and create many different facial expressions. An example is if the eyes are closed or the mouth is smiling on the test image, this has severe effect on recognition if the trained database images have open eyes and neutral mouth. Since dynamic features are difficult to handle in static images one solution is therefore to exclude the bottom part of the face as showed in the facenormtop module.

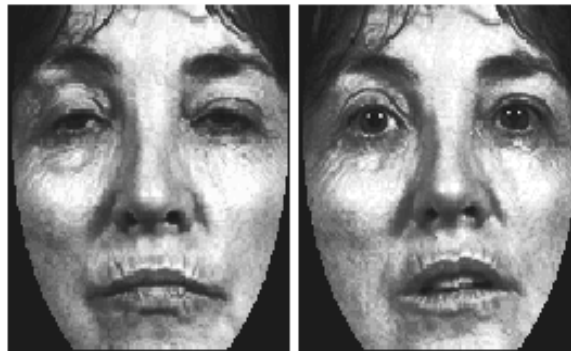


Figure 45: Example of candidate with closed eyes and different mouth expression

My first experiment was therefore to look at the difference in recognition rate between these two normalizations (Figure 46). This was done using the testgabor module and examining the sorted candidate vectors. A rectangular sampling space, 20 pixels wide, was used both for full and top normalization. All 3816 frontal images from COORDS3816.TXT were used. The columns show the recognition rates for full head normalization (red) and top head normalization (yellow). These first two columns examine only the best candidate, which would be the final answer if this was not a hybrid recognizer. The next three groups of columns examine the 1%, 5% or 10% best candidates of the sorted candidate vector. This means that the recognition rates show the certainty that a correct answer exists among the promising candidates if 1%, 5% or 10% of the best candidates are sent to fine secondary screening.

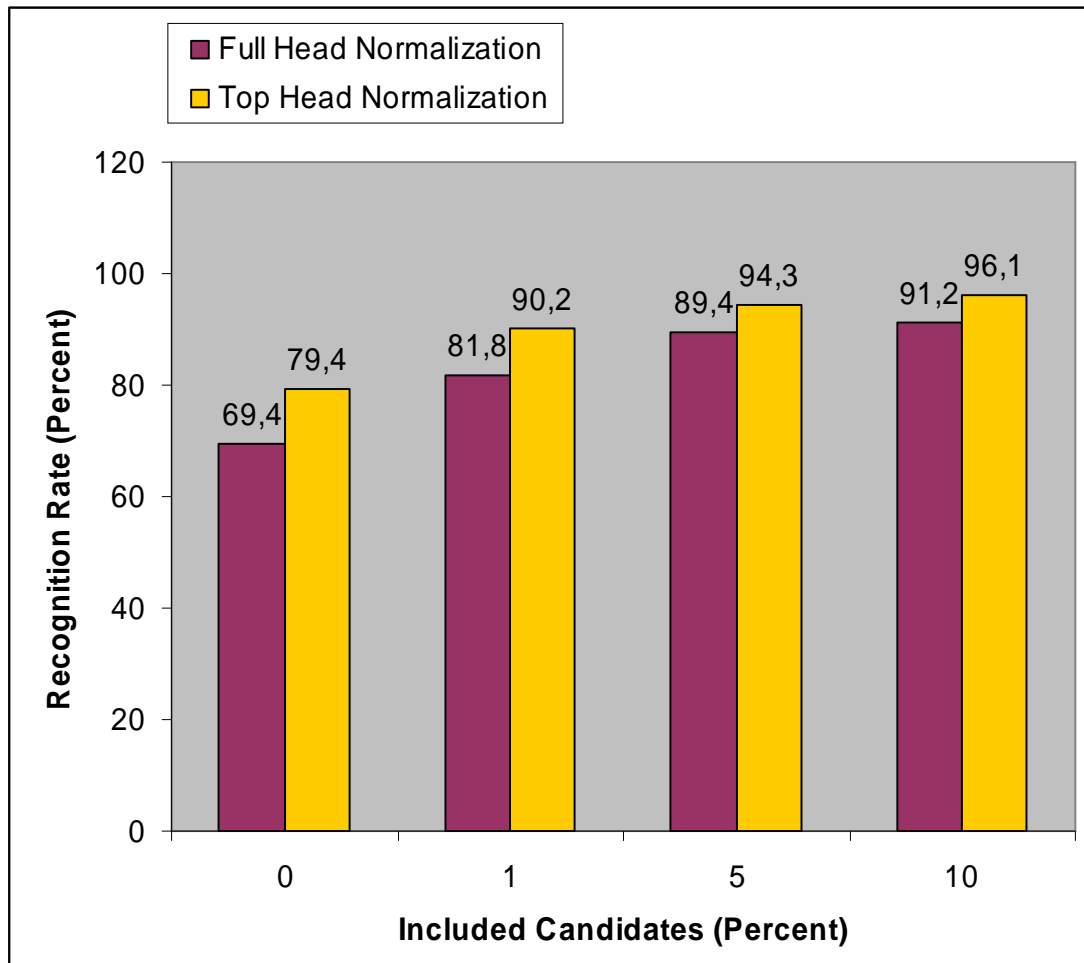


Figure 46: Comparison of full against top head normalization

The results show that normalization gains about 10% in overall recognition rate when using the top of the head only. When as much as 10% of the best candidates are sent away up to 5% is gained in certainty that the correct answer is among the candidates. This should be a clear indication for preferring top head normalization for use with the Gabor recognizer.

## 7.2 How Many Gabor Jet Samples in the Rectangular Grid?

Another interesting question was what is the optimal rectangular grid space used for sampling. This second experiment used top head normalized images with a rectangular sampling space of 10, 20 or 40 pixels (Figure 47). A sampling space of 10 pixels between each sample results in 130 Gabor jets (blue column), 20 pixels results in 35 Gabor jets (red) and 40 pixels results in 12 Gabor jets (yellow). Also here all the 3816 frontal images from COORDS3816.TXT were used.

Also here we can see that taking into account more candidates gives larger certainty that a correct candidate is among the promising candidates sent to secondary fine screening. Another fact is that decreasing the sampling space from 40 to 20 pixels gives an increased recognition rate of about 9%. What is more interesting is that decreasing the sampling space from 20 pixels to 10 pixels gives no gained recognition

rate. In fact the recognition rate has dropped a little bit. This assures that choosing a sampling space of 20 pixels giving a total of 35 Gabor jets is a good choice. Increasing the pixel space will reduce the recognition performance and decreasing the pixel space will have a bad effect on the total running time as well as having no effect on recognition performance.

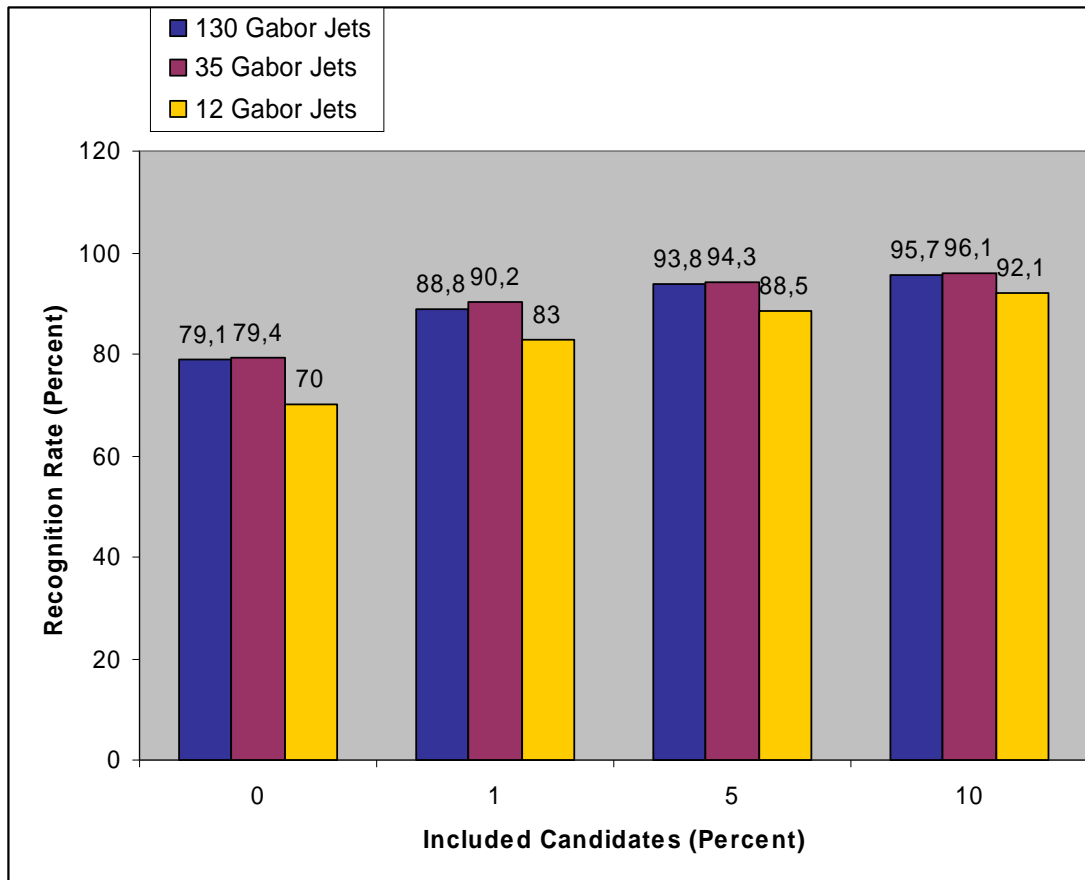


Figure 47: Comparison of 10, 20 and 40 rectangular pixel spacing

### 7.3 Tuning the Gabor module

Ideally the certainty for having a correct candidate among the promising candidates should be 100%. By tuning the Gabor module I finally managed to get this as high as 98.7%. This means that some correct candidates unfortunately are ruled out before they reach a final decision (that is about 13 of 1000). One thing I found out during my investigation of what went wrong was that the eye coordinates in the coordinate file supplied with the FERET database are not always that well placed. The Gabor module is very sensitive to even minor vertical and horizontal translations. It is therefore very important that the eye coordinates of the same individual are placed on the same spot. To improve the recognition results I changed the eye coordinates for some bad cases and tried to place the two spots exactly in the middle of the eye pupils. Closed eyes are of course a problem and the two spots were in these cases just roughly placed in the right position.

## 7.4 L2norm, Mahalanobis or a Combined Distance Measure?

Finally let us see how well the hybrid combination performs combining the strengths of both Gabor coarse screening and PCA fine screening. Originally I wanted to use the regular L2norm as the PCA classifier. I also wanted to compare L2norm performance with the more advanced Mahalanobis distance which takes into account the axis eigenvalues as weighting. Other classifiers like city block distance or chessboard are not equally interesting because they do not include the eigenvalues in the calculation.

My results showed that the L2norm classifier is in fact better than Mahalanobis distance (Figure 48). However I noticed one remarkable feature with the Mahalanobis distance. Almost every false recognized picture got a distance in the range  $-10^5$  to  $-10^7$  while correctly recognized pictures got a distance in the range  $-10^0$  to  $-10^1$ . This means that the recognition rate of 76% can be increased drastically by reprocessing all pictures with a Mahalanobis distance of less than a score of about -1000. My solution was to apply a regular L2norm to these false candidates, which raised the recognition rate considerably to 90% correct. This combined classifier is a good example of a hybrid combination where the individual properties of Mahalanobis distance and L2norm complement each other.

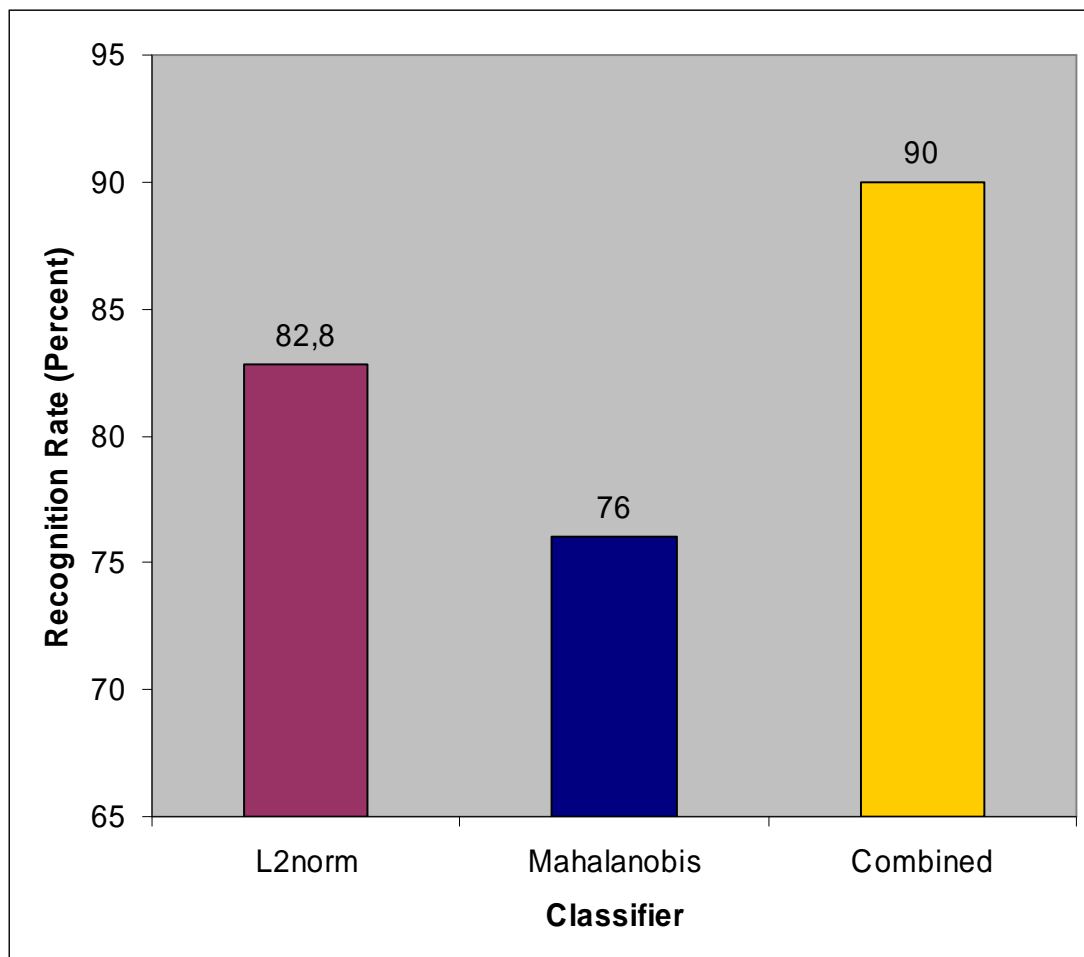


Figure 48: Comparison of using L2norm, Mahalanobis distance and a combined classifier

## 7.5 Head Rotated Face Images

When it comes to head rotated images the result was a recognition rate of 64,1%. This is okay, not as good as expected, and clearly not good enough. Head rotated face images seem like a quite more difficult topic than frontal images. I suggest that more effort should be put in the normalizing module with finding the correct rotation angle of the face.

## 7.6 Elapsed Time

Time elapsed during different steps is of course very dependent on the computer which was used. I have been using a standard personal computer, with a 2.4 GHz Intel Pentium 4 processor and 512 MB of DDR memory.

Hybrid training, which consists of normalization and a lot of Gabor wavelet convolutions, of one image takes about 0.75 seconds. Normalizing one image takes about 0.30 seconds and Gabor training takes about 0.45 seconds. As an example it takes approximately 5 minutes to train all the 423 head rotated images (2 minutes normalizing and 3 minutes convoluting).

Hybrid recognition takes much more time, but luckily in most cases we only want to recognize one image quickly. Total time elapsed for recognizing one image is about 25 seconds. This looks like an acceptable time. During testing I have compared each image in the database against all the other images. As an example recognizing 1000 images equals a total of 7 hours computer work.

Training Time (whole database – N images) = 0,75s \* N

Add Image to Database (one image) = 0,75s

Recognition Time (one image) = 25 s

## 8 Conclusion

Some conclusions can be drawn. The most important one is that hybrid systems are important for future advances in automated face recognition [117][118]. My results showed a 20% increase in overall recognition rate between a plain Gabor wavelet implementation and a hybrid combination of Gabor wavelets and PCA. The recognition rate is of course very dependent on the actual implementation and the quality of the face images.

Using Gabor wavelets in face recognition is biological motivated and seems like a very interesting approach not only to face recognition, but machine recognition in general. Gabor jet images combined with a size, position and angle accurate normalization process shows good recognition performance. Even small variances in location of feature points of trained and test images however can have a bad effect on the recognition performance.

The drawbacks are compensated with regular PCA. One major advantage with this hybrid combination is that it provides methods for incremental learning of new classes. This means that new faces can be added to the database without recomputing the representations of all other learned faces. The final classification was done using a combination of Mahalanobis distance and L2norm. This is another example which shows that a hybrid combination can outperform the two individual techniques.

Using hybrid methods can be seen as using an appropriate compressed feature representation and having a good classification scheme for separating the data into the correct classes. The fine and course sorting mechanisms help to reduce computation time by reducing the number of possible candidates. Different sorting techniques have been little discussed in previous face recognition literature, but they certainly play an important part when dealing with large face databases.

All frontal images were recognized with 90% certainty, which is quite satisfying. Recognizing rotated head images was more difficult and resulted in 64% correct only and obviously requires more investigation. The normalization needs to know the head rotation angle quite accurately to be able to perform well. It is clear that the primitive method of finding an approximate value was not good enough.



## 9 References

### 9.1 Litterature

#### Introduction

- [1] “Face Recognition Using Gabor Wavelets”, Atle Nes (2002)
- [2] “Face Recognition Using Eigen Light Fields”, Håkon Heuch (2002)
- [3] “Ansiksgjenkjenning”, Lars Tellemann Sæther (2002)
- [4] “Introduction to Face Recognition”, A. Jonathan Howell (1999)

#### Templates

- [5] “Picture processing by computer complex and recognition of human faces”, T. Kanade (1973)
- [6] “Feature Extraction from Faces Using Deformable Templates”, Alan L. Yuille, David S. Cohen and Peter W. Hallianan (1989)
- [7] “Face Recognition: Features Versus Templates”, Roberto Brunelli and Tomaso Poggio (1993)

#### PCA :: Eigenfaces

- [8] ”Low-Dimensional Procedure for the Characterization of Human Faces”, L. Sirovich and M. Kirby (1987)
- [9] “Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces”, M. Kirby and L. Sirovich (1990)
- [10] “Eigenfaces for Recognition”, Matthew Turk, Alex Pentland (1991)
- [11] “Face Recognition Using Eigenfaces”, Matthew A. Turk and Alex P. Pentland (1991)
- [12] “View-Based and Modular Eigenspaces for Face Recognition”, Alex Pentland, Baback Moghaddam and Thad Starner (1994)
- [13] “A Random Walk Trough Eigenspace”, Matthew Turk (2001)
- [14] “Subspace Methods for Robot Vision”, Shree K. Nayar, Sameer A. Nene and Hiroshi Murase (1995)
- [15] “Visual Learning and Recognition of 3D Objects From Appearance”, H. Murase and S. Nayar (1995)

- [16] “Colour Eigenfaces”, Graham D. Finlayson, Janet Dueck, Brian V. Funt, Mark S. Drew (1996)
- [17] “Beyond Euclidean Eigenspaces: Bayesian Matching for Visual Recognition”, Baback Moghaddam, Alex Pentland (1998)

### **PCA :: Eigenfeatures**

- [18] “Recognizing Faces from the Eyes Only”, E. Hjelmås, J. Wroldsen (1999)
- [19] “Eigenfaces versus Eigeneyes: First Steps Towards Performance Assessment of Representations for Face Recognition”, Teófilo Emídio Campos, Rogério Schmidt Feris and Roberto Marcondes Cesar Junior (2000)

### **PCA :: Eigen Light-Fields**

- [20] “Eigen Light-Fields and Face Recognition Across Pose”, Ralph Gross, Iain Matthews and Simon Baker (2002)
- [21] “Appearance-Based Face Recognition and Light-Fields”, Ralph Gross, Iain Matthews and Simon Baker (2002)
- [22] “Combining Models and Exemplars for Face Recognition: An Illuminating Example”, Terence Sim and Takeo Kanade (2001)
- [23] “When is the Shape of a Scene Unique Given its Light-Field: A Fundamental Theorem of 3D Vision?”, Simon Baker, Terence Sim and Takeo Kanade (2002)
- [24] “What is the Set of Images of an Object Under All Possible Lighting Conditions”, Peter N. Belhumeur and David J. Kriegman (1996)
- [25] “From Few to Many: Illumination Cone Models for Face Recognition Under Variable Lightning and Pose”, Athinodoros S. Georghiadis and Peter Belhumeur (2000)
- [26] “The Plenoptic Function and the Elements of Early Vision”, Edward H. Adelson and James R. Bergen (1991)
- [27] “Light Field Rendering”, Marc Levoy and Pat Hanrahan (1996)
- [28] “Dealing with Occlusions in the Eigenspace Approach”, Ales Leonardis and Horst Bischof (1996)
- [29] “Robust Recognition Using Eigenimages”, Ales Leonardis, Horst Bischof and Roland Ebensberger (1997)

**PCA :: Fourier Domain**

- [30] “A Robust Face Identification Scheme – KL Expansion of an Invariant Feature Space”, H.F.S. Akamatsu, T. Sasaki and Y. Suenuga (1991)

**PCA :: Evaluation**

- [31] “Experiments on Eigenfaces Robustness”, Alexandre Lemieux and Marc Parizeau (2002)
- [32] “Analysis of PCA-Based and Fisher Discriminant-Based Image Recognition Algorithms”, Wendy S. Yambor (2000)
- [33] “Analyzing PCA-Based Face Recognition Algorithms: Eigenvector Selection and Distance Measures”, Wendy S. Yambor, Bruce A. Draper and J. Ross Beveridge (2000)
- [34] “A Nonparametric Statistical Comparison of Principal Component and Linear Discriminant Subspaces for Face Recognition”, J. Ross Beveridge, Kai She and Geof H. Givens (2001)
- [35] “The Global Dimensionality of Face Space”, P. S. Penev and L. Sirovich (2000)
- [36] “Numerical Analysis of Eigenfaces”, Neil Muller and B. M. Herbst (2000)
- [37] “Learning in Eigenspace: Theory and Application”, B.S. Manjunath, S. Chandrasekaran, Y.F. Wang (1994)
- [38] “An Eigenspace Update Algorithm for Image Analysis”, S. Chandrasekaran, B.S. Manjunath, Y.F. Wang, J. Winkeler and H. Zhang (1996)

**LDA :: Fisherfaces**

- [39] “The Use of Multiple Measurements in Taxonomic Problems”, R. A. Fisher (1936)
- [40] “Using Discriminant Eigenfeatures for Image Retrieval”, D. L. Swets and J. Weng (1996)
- [41] “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection”, Peter N. Belhumeur, Joao P. Hespana and David J. Kriegman (1996)
- [42] “Discriminant Analysis of Principal Components for Face Recognition”, W. Zhao, A. Krishnaswamy, R. Chellappa, D. L. Swets and J. Weng (1998)
- [43] “Discriminant Analysis for Recognition of Human Face Images”, K. Etemad and R. Chellappa (1997)

- [44] “Face Recognition Algorithms Review”, Sunny Tang Ho Man (2001)

### **LDA :: Fisher Light-Fields**

- [45] “Fisher Light-Fields for Face Recognition Across Pose and Illumination”, Ralph Gross, Ian Matthews and Simon Baker (2002)

### **LDA :: Evaluation**

- [46] “Analysis and Comparison of Eigenspace-Based Face Recognition Approaches”, Pablo Navarrete and Javier Ruiz-del-Solar (2002)

### **ICA :: Introduction**

- [47] “Face Recognition Using Independent Component Analysis and Support Vector Machines”, O. Déniz, M. Castrillón and M. Hernández (2001)
- [48] “Independent Component Analysis: A Tutorial”, Aapo Hyvärinen and Erkki Oja (1999)
- [49] “Viewpoint Invariant Face Recognition Using Independent Component Analysis and Attractor Networks”, Marian Stewart Bartlett and Terrence J. Sejnowski (1997)
- [50] “Independent Components of Face Images: A Representation for Face Recognition”, Marian Stewart Bartlett and Terrence J. Sejnowski (1997)
- [51] “Independent Component Representations for Face Recognition”, Marian Stewart Bartlett, H. Martin Lades and Terrence J. Sejnowski (1998)

### **ICA :: Tensorfaces**

- [52] “Multilinear Analysis of Image Ensembles: Tensorfaces”, M. Alex O. Vasilescu and Demetri Terzopoulos (2002)
- [53] “Multilinear Image Analysis for Facial Recognition”, M. Alex O. Vasilescu and Demetri Terzopoulos (2002)

### **ICA :: Evaluation**

- [54] “Comparative Assessment of Independent Component Analysis (ICA) for Face Recognition”, Chengjun Liu and Harry Wechsler (1999)
- [55] “PCA vs. ISA: A Comparison on the FERET Data Set”, Kyungim Baek, Bruce A. Draper, J. Ross Beveridge and Kai She (2001)

### **Wavelets :: Gabor Wavelets**

- [56] “Mathematical Description of the Responses of Simple Cortical Cells”, S. Marcelja (1980)

- [57] “Theory of Communication”, Dennis Gabor (1946)
- [58] “Two Dimensional Spectral Analysis of Cortical Receptive Field Profile”, J.G. Daugman (1980)
- [59] “A Feature Based Approach to Face Recognition”, B.S. Manjunath, R. Chellappa and Christoph von der Malsburg (1992)
- [60] “Face Recognition by Elastic Bunch Graph Matching”, Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger and Christoph von der Malsburg (1996)
- [61] “Distortion Invariant Object Recognition in the Dynamic Link Architecture”, Martin Lades, Jan C. Vorbrüggen, Joachim Buhmann, Jörg Lange, Christoph von der Malsburg, Rolf P. Würtz and Wolfgang Konen (1993)
- [62] “Face Recognition by Eigenface and Elastic Bunch Graph Matching”, Jang Kim Fung (1999)
- [63] “Face Recognition Using Gabor Wavelet Transform”, Burcu Kepenekci (2001)
- [64] “Biometric Systems: A Face Recognition Approach”, Erik Hjelmås (2000)
- [65] “Feature-Based Face Recognition”, Erik Hjelmås (2000)

#### **Wavelets :: Gabor Feature Classifier**

- [66] “A Gabor Feature Classifier for Face Recognition”, Chenjun Liu and Harry Wechsler (2001)
- [67] “Gabor Feature Based Classification Using the Enhanced Fisher Linear Discriminant Model for Face Recognition”, Chenjun Liu and Harry Wechsler (2002)

#### **Wavelets :: Evaluation**

- [68] “Face Recognition: Graph Matching Versus Neural Techniques”, J. Vergés-Llahí, A. Sanfeliu, F. Serratos, R. Alquezar (1999)

#### **Hidden Markov Models**

- [69] “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”, Lawrence R. Rabiner (1989)
- [70] “Parameterization of a Stochastic Model for Human Face Identification”, F. S. Samaria and A. C. Harter (1994)
- [71] “Face Recognition Using Hidden Markov Models, Ferdinando Silvestro Samaria (1994)

- [72] “Statistical Approaches to Face Recognition”, Ara V. Nefian (1996)
- [73] “Hidden Markov Models for Face Recognition”, Ara V. Nefian and Monson H. Hayes III (1996)
- [74] “Face Detection and Recognition Using Hidden Markov Models”, Ara V. Nefian and Monson H. Hayes III (1998)
- [75] “A Hidden Markov Model-Based Approach for Face Detection and Recognition”, Ara V. Nefian (1998)
- [76] “Face Recognition Using an Embedded HMM”, Ara V. Nefian and Monson H. Hayes III (1998)
- [77] “An Embedded HMM-Based Approach for Face Detection and Recognition”, Ara V. Nefian and Monson H. Hayes III (1998)
- [78] “Maximum Likelihood Training of the Embedded HMM for Face Detection and Recognition”, Ara V. Nefian and Monson H. Hayes III (2000)
- [79] “High Performance Face Recognition Using Pseudo 2-D Hidden Markov Models”, Stefan Eickeler, Stefan Müller and Gerhard Rigoll (1998)
- [80] “Improved Face Recognition Using Pseudo 2-D Hidden Markov Models”, Stefan Eickeler, Stefan Müller and Gerhard Rigoll (1998)
- [81] “Face Database Retrieval Using Pseudo 2D Hidden Markov Models”, Stefan Eickeler (2002)
- [82] “Low Complexity 2-D Hidden Markov Model for Face Recognition”, H. Othman and T. Aboulnasr (2000)
- [83] “A Simplified Second-Order HMM with Application to Face Recognition”, H. Othman and T. Aboulnasr (2001)
- [84] “Standard Support for Automatic Face Recognition”, Ara Nefian and Bob Davies (2001)

### **Neural Networks**

- [85] “Biological Inspired Face Detection and Recognition Using Spiking Neurons”, Vishal Vaingankar and Roger S. Gaborski (2002)

### **Neural Networks :: Multi-layer Perceptron (MLP)**

- [86] “Access Control by Face Recognition Using Neural Networks”, Dmitry Bryliuk and Valery Starovoitov (2000)

- [87] “Face Recognition Using Ensembles of Networks”, S. Gutta, J. Huang, B. Takacs and H. Wechsler (1996)
- [88] “Pose Invariant Face Recognition”, Fu Jie Huang, Zhihua Zhou, Hong-Jiang Zhang and Tsuhan Chen (2000)
- [89] “Multilayer Perceptrons versus Hidden Markov Models: Comparisons and Applications to Image Analysis and Visual Pattern Recognition”, Lúcio F.C. Pessoa (1995)
- [90] “Face Recognition Using Unsupervised Feature Extraction”, G. W. Cottrell and M. Flemming (1990)
- [91] “Face Recognition: A Hybrid Neural Network Approach”, Steve Lawrence, C. Lee Giles, Ah Chung Tsoi and Andrew D. Baek (1996)
- [92] “Face Recognition: A Convolutional Neural Network Approach”, Steve Lawrence, C. Lee Giles, Ah Chung Tsoi and Andrew D. Baek (1997)
- [93] “Face Recognition with Multi-Layer Perceptrons”, Erik Hjelmås and Jørn Wroldsen (1997)
- [94] “Self-Organizing Map”, T. Kohonen (1990)

#### **Neural Networks :: Radial Basis Function (RBF)**

- [95] “Receptive Field Functions for Face Recognition”, A. Jonathan Howell and Hillary Buxton (1995)
- [96] “Face Recognition Using Radial Basis Function Neural Networks”, A. Jonathan Howell and Hillary Buxton (1996)
- [97] “Towards Unconstrained Face Recognition from Image Sequences”, A. Jonathan Howell and Hillary Buxton (1996)
- [98] “Automatic Face Recognition Using Radial Basis Functions”, Andrew Jonathan Howell (1997)
- [99] “Face Unit Radial Basis Function Networks”, A. Jonathan Howell (1999)
- [100] “Face Recognition Using Radial Basis Function (RBF) Neural Networks”, Meng Joo Er, Shiqian Wu and Juwei Lu (1999)
- [101] “Face Recognition With Radial Basis Function (RBF) Neural Networks, Meng Joo Er, Shiqian Wu, Juwei Lu and Hock Lye Toh (2002)
- [102] “Comparing the Performance of the Discriminant Analysis and RBF Neural Network for Face Recognition”, Raul Queiroz Feitosa (1999)

- [103] “An Experimental Study: On Reducing RBF Input Dimension by ICA and PCA”, Rong-Bo Huang, Lap-Tak Law and Yiu-Ming Cheung (2002)
- [104] “Face Recognition Using Hybrid Classifier Systems”, Srinivas Gutta and Harry Wechsler (1999)

### **Neural Networks :: Dynamic Link Architecture (DLA)**

- [105] “Face Recognition by Dynamic Link Matching”, Laurenz Wiskott and Christoph von der Malsburg (1999)

### **Hybrid**

- [106] “Face Recognition: a Summary of 1995 - 1997”, Thomas Fromherz (1998)
- [107] “Face Recognition from Frontal and Profile Views”, Gaile G. Gordon (1995)
- [108] “Fusion of LDA and PCA for Face Recognition”, Gian Luca Marcialis and Fabio Roli (2002)
- [109] “Combination of Classifiers on the Decision Level for Face Recognition”, Bernhard Acherhmann and Horst Bunke (1996)
- [110] “Information Fusion and Person Identification Using Speech & Face Information”, Conrad Sanderson (2002)
- [111] “Illumination Invariant Face Recognition Using Thermal Imagery”, Diego A. Socolinsky, Lawrence B. Wolff, Joshua D. Neuheisel and Christopher K. Eveland (2001)
- [112] “Appearance-Based Facial Recognition Using Visible and Thermal Imagery: A Comparative Study”, Andrea Selinger and Diego A. Socolinsky (2001)
- [113] “Face Recognition in Thermal Infrared”, Lawrence B. Wolff, Diego A. Socolinsky and Christopher K. Eveland (2001)
- [114] “Face and Hand Gesture Recognition Using Hybrid Classifiers”, Srinivas Gutta, Jeffrey Huang, Ibrahim F. Imam and Harry Wechsler (1996)
- [115] “A Novel Hybrid Face Profile Recognition System Using the FERET and Mugshot Databases”, Frank Wallhoff and Gerhard Rigoll (2001)
- [116] “Hybrid Face Recognition Systems for Profile Views Using the Mugshot Database”, Frank Wallhoff, Stefan Müller and Gerhard Rigoll (2001)

### **Hybrid :: Conclusion**

- [117] “Benchmark Studies on Face Recognition”, Srinivas Gutta, Jeffrey Huang, Imran Shah, Barnabás Tackács and Harry Wechsler (1995)



[118] “A Survey of Face Recognition Algorithms and Testing Results”, William A. Barrett (1998)

## 9.2 *Hyperlinks*

Evaluation of Face Recognition Algorithms : <http://www.cs.colostate.edu/evalfacerec/>  
Face Recognition Vendor Test : <http://www.frvt.org/>

Thomas Fromherz homepage : <http://www.fromherz.net/>  
Gaile G. Gordon's homepage : <http://www.vincent-net.com/gaile/>  
Volker Blanz homepage : <http://www.mpi-sb.mpg.de/~blanz/>

Intel Open Source Computer Vision Library :  
<http://www.intel.com/research/mrl/research/opencv/>  
Mathworks : <http://www.mathworks.com/>

## 10 Indexes

### 10.1 Figures

FIGURE 1: TYPICAL EDGE PROJECTIONS DATA .....	14
FIGURE 2: EDGE DOMINANCE MAPS (A) HORIZONTAL (LEFT) (B) VERTICAL (RIGHT) .....	14
FIGURE 3: DYNAMIC SEQUENCE FOR EYE DETECTION .....	15
FIGURE 4: PRINCIPAL COMPONENT ANALYSIS (PCA) .....	17
FIGURE 5: EIGENFACES HAVE A FACE-LIKE APPEARANCE .....	18
FIGURE 6: EIGENSPACE RECOGNITION (A) APPEARANCE MODEL (LEFT) (B) DISCRIMINATIVE MODEL (RIGHT) .....	19
FIGURE 7: SIMPLIFIED EXAMPLE OF A 2D LIGHT-FIELD FROM A 2D OBJECT .....	20
FIGURE 8: (A) 3-POINT NORMALIZATION (LEFT) (B) MULTI-POINT NORMALIZATION (RIGHT) .....	21
FIGURE 9: (A) POINTS IN TWO-DIMENSIONAL SPACE (B) POOR SEPARATION (C) GOOD SEPARATION .....	23
FIGURE 10: ALGORITHM COMPARISON (A) ACROSS POSE (LEFT) (B) ACROSS POSE AND ILLUMINATION (RIGHT) .....	24
FIGURE 11: ICA IMAGE SYNTHESIS MODEL .....	26
FIGURE 12: ICA ON IMAGES WITH SOURCE SEPARATION ON (A) FACE IMAGES (B) FACE PIXELS .....	27
FIGURE 13: 3D REPRESENTATION OF A GABOR WAVELET IN (A) SPACE DOMAIN (B) FREQUENCY DOMAIN .....	29
FIGURE 14: (A) ENSAMBLE OF GABOR WAVELETS (B) COVERAGE OF SPATIAL FREQUENCY PLANE .....	30
FIGURE 15: (A) ORIGINAL IMAGE (B) GABOR WAVELETS (C) CONVOLUTION RESULT (D) JET .....	30
FIGURE 16: GRAPHS FOR FACES IN DIFFERENT VIEWS .....	31
FIGURE 17: FACE BUNCH GRAPH WITH LOCAL EXPERTS AT EACH FIDUCIAL POINT MARKED GREY .....	32
FIGURE 18: LEFT-TO-RIGHT STATES OF A ONE-DIMENSIONAL HMM .....	35
FIGURE 19: IMAGE SAMPLING TECHNIQUE FOR ONE-DIMENSIONAL HMM .....	35
FIGURE 20: STATES OF A PSEUDO TWO-DIMENSIONAL HMM .....	36
FIGURE 21: IMAGE SAMPLING TECHNIQUE FOR PSEUDO TWO-DIMENSIONAL HMM .....	36
FIGURE 22: HMM TRAINING SCHEME .....	37
FIGURE 23: HMM RECOGNITION SCHEME .....	38
FIGURE 24: FEED-FORWARD NEURAL NETWORK .....	40
FIGURE 25: NEURAL NETWORK FACE RECOGNITION SYSTEM .....	41
FIGURE 26: FRONTAL AND PROFILE TEMPLATES FOR HYBRID FACE RECOGNITION .....	45
FIGURE 27: HYBRID FUSION OF PCA AND LDA .....	45
FIGURE 28: HYBRID PARALLEL COMBINATION SCHEME .....	46
FIGURE 29: ERBF1 (LEFT) AND ERBF2 (RIGHT) .....	47
FIGURE 30: HYBRID SERIAL APPROACH WITH A NEURAL NETWORK AND HMM .....	48
FIGURE 31: INTUITIVE NEW WAY OF LOOKING ON THE PROBLEM OF HYBRID FACE RECOGNITION .....	49
FIGURE 32: FLOW CHART OF THE NORMALIZATION MODULE .....	50
FIGURE 33: FLOW CHART OF THE COARSE PRIMARY SCREENING MODULE .....	50
FIGURE 34: FLOW CHART OF FINE SECONDARY SCREENING MODULE .....	52
FIGURE 35: MATLAB GRAPHICAL USER INTERFACE AND RUNNING A SIMPLE PROGRAM .....	54
FIGURE 36: HORIZONTAL ALIGNMENT OF THE RIGHT AND LEFT EYE BY ROTATING THE IMAGE .....	56
FIGURE 37: FINDING THE NEW EYE COORDINATES IN THE ROTATED IMAGE .....	56
FIGURE 38: CROPPING THE IMAGE USING EYEROW AND EYEDISTANCE .....	57
FIGURE 39: EXAMPLE NORMALIZATION OF A FACE IMAGE .....	58
FIGURE 40: EXAMPLE NORMALIZATION OF THE TOP OF THE FACE .....	58
FIGURE 41: ROTATED FACE SEEN FROM ABOVE .....	59
FIGURE 42: TWO NORMALIZED HEAD ROTATED IMAGE SERIES AND THEIR CORRESPONDING ROTATION ANGLES .....	61
FIGURE 43: CONVOLUTION WITH 45 DEGREE EVEN MASKS WITH DIFFERENT MASK SIZES (A) MASK1 11x11 PIXELS (B) MASK5 23x23 PIXELS (C) MASK9 47x47 PIXELS (D) MASK13 95x95 PIXELS .....	62
FIGURE 44: CONVOLUTION WITH EVEN MASKS OF SIZE 11x11 PIXELS WITH DIFFERENT ROTATIONS (A) MASK0 0 DEGREES (B) MASK1 45 DEGREES (C) MASK2 90 DEGREES (D) MASK3 135 DEGREES .....	63
FIGURE 45: EXAMPLE OF CANDIDATE WITH CLOSED EYES AND DIFFERENT MOUTH EXPRESSION .....	67
FIGURE 46: COMPARISON OF FULL AGAINST TOP HEAD NORMALIZATION .....	68
FIGURE 47: COMPARISON OF 10, 20 AND 40 RECTANGULAR PIXEL SPACING .....	69
FIGURE 48: COMPARISON OF USING L2NORM, MAHALANOBIS DISTANCE AND A COMBINED CLASSIFIER .....	70

## 11 Appendix

The supplied Matlab code should be useful for further investigations into and improvements to hybrid face recognition. This chapter contains source code from the most important files.

### 11.1 Facenorm.m

```
function [] = facenorm(number)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dataPath = ['.\\DATA\\'];
coordFile = ['COORDS3816.TXT'];
maskFile = ['M150X130.DAT'];

imagePath = ['.\\FERET-PGM\\'];
imageFormat = ['.pgm'];

normPath = ['.\\NORM\\'];
normFormat = ['.nrm.pgm'];

eyedistance = 70;
eyerow = 45;
norm_height = 150;
norm_width = 130;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
data = [dataPath coordFile];
[file, xeyeleft, yeyeleft, xeyeright, yeyeright] = textread(data, '%s %d %d %d %d %d %d %d %d %d');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = number;

imageName = char(file(n));
name = [imagePath imageName imageFormat];
im = imread(name);
[im_h,im_w] = size(im);

fprintf('imagename: %s \n', imageName);

fprintf('Image Height: %d \n', im_h);
fprintf('Image Width: %d \n', im_w);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hdiff = yeyeright(n) - yeyeleft(n);
wdiff = abs(xeyeright(n) - xeyeleft(n));
angle_rad = atan(hdiff/wdiff);
angle_deg = angle_rad*(180/pi);
im_rotated = imrotate(im,angle_deg);

fprintf('Eye hdiff: %d \n', hdiff);
fprintf('Eye wdiff: %d \n', wdiff);
fprintf('Rotation Angle: %d \n', angle_deg);
fprintf('Size Rotated: %d\n\n', size(im_rotated));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if angle_deg >= 0
    xeyeleft_new = xeyeleft(n)*cos(angle_rad) + yeyeleft(n)*sin(angle_rad);
    yeyeleft_new = (im_w-xeyeleft(n))*sin(angle_rad) + yeyeleft(n)*cos(angle_rad);
    xeyeright_new = xeyeright(n)*cos(angle_rad) + yeyeright(n)*sin(angle_rad);
    yeyeright_new = (im_w-xeyeright(n))*sin(angle_rad) + yeyeright(n)*cos(angle_rad);
else
    xeyeleft_new = xeyeleft(n)*cos(-angle_rad) + (im_h-yeyeleft(n))*sin(-angle_rad);
    yeyeleft_new = xeyeleft(n)*sin(-angle_rad) + yeyeleft(n)*cos(-angle_rad);
    xeyeright_new = xeyeright(n)*cos(-angle_rad) + (im_h-yeyeright(n))*sin(-angle_rad);
    yeyeright_new = xeyeright(n)*sin(-angle_rad) + yeyeright(n)*cos(-angle_rad);
end
```

```

scale = eyedistance/(xeyeright_new-xeyeleft_new);

fprintf('New Left Eye Height: %d \n', yeyeleft_new);
fprintf('New Left Eye Width: %d \n', xeyeleft_new);
fprintf('New Right Eye Height: %d \n', yeyeright_new);
fprintf('New Right Eye Width: %d \n', xeyeright_new);
fprintf('Scale Factor: %d \n', scale);

im_resized = imresize(im_rotated,scale);

fprintf('Size Resized: %d\n\n', size(im_resized));

%%%%%%%%%% Crop Image %%%%%%%%%%%
x1 = floor( xeyeleft_new * scale - (norm_width - eyedistance)/2);
y1 = floor( yeyeleft_new * scale - eyerow );

im_cropped = imcrop(im_resized,[x1 y1 norm_width-1 norm_height-1]);

fprintf('New2 Left Eye Height: %d \n', y1);
fprintf('New2 Left Eye Width: %d \n', x1);
fprintf('Size: %d', size(im_cropped));

%%%%%%%%%% Convolute Mask With The Image %%%%%%%%%%%
fid = fopen([dataPath maskFile]);
mask = fread(fid);
fclose(fid);

im_mask = im_cropped;

for i=1:norm_height
    for j=1:norm_width
        if mask((i-1)*norm_width+j) == 0
            im_mask(i,j) = 0;
        end
    end
end

%%%%%%%%%% Histogram Equalize Image %%%%%%%%%%%
im_hist = histeq(im_mask);

%%%%%%%%%% Plot Normalization Steps %%%%%%%%%%%
subplot(2,2,1);imshow(im); title('Original');
subplot(2,2,2);imshow(im_rotated); title('Rotated');
subplot(2,2,3);imshow(im_cropped); title('Cropped');
subplot(2,2,4);imshow(im_hist); title('Equalized');
pause; close;

%%%%%%%%%% Write Normalized Image %%%%%%%%%%%
imwrite(im_hist,[normPath imageName normFormat]);

```

## 11.2 Facenormrot.m

```

function [] = facenormrot(number)

%%%%%%%%%% Global Definitions %%%%%%%%%%%
dataPath = ['. \DATA\'];
coordFile = ['ROTATED.TXT'];
maskFile = ['M150X130.DAT'];

imagePath = ['. \FERET-PGM\'];
imageFormat = ['.pgm'];

normPath = ['. \NORMROT\'];
normFormat = ['.nrm.pgm'];

eyedistance = 70;
eyerow = 45;
norm_height = 150;
norm_width = 130;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Collecting Image Data from datafile %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
data = [dataPath coordFile];
[file, xeyeleft, yeyeleft, xeyeright, yeyeright] = textread(data, '%s %d %d %d %d %d %d %d %d %d');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Choose Rotated Image Serie %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if number == 0
    start = 1; stop = size(file); % Normalize all pictures
else
    start = 9*(number-1)+1;
    stop = 9*(number-1)+9;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Read Images To Be Normalized %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plotnumber = 1;

for n=start:stop

    imageName = char(file(n));
    name = [imagePath imageName imageFormat];
    im = imread(name);
    [im_h,im_w] = size(im);

    fprintf('imagename: %s \n', imageName);

    %fprintf('Image Height: %d \n', im_h);
    %fprintf('Image Width: %d \n', im_w);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Rotate Image %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    hdiff = yeyeright(n) - yeyeleft(n);
    wdiff = abs(xeyeright(n) - xeyeleft(n));
    angle_rad = atan(hdiff/wdiff);
    angle_deg = angle_rad*(180/pi);
    im_rotated = imrotate(im,angle_deg);

    %fprintf('Eye hdiff: %d \n', hdiff);
    %fprintf('Eye wdiff: %d \n', wdiff);
    %fprintf('Rotation Angle: %d \n', angle_deg);
    %fprintf('Size Rotated: %d\n\n', size(im_rotated));

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Resize Image %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if angle_deg >= 0
        xeyeleft_new = xeyeleft(n)*cos(angle_rad) + yeyeleft(n)*sin(angle_rad);
        yeyeleft_new = (im_w-xeyeleft(n))*sin(angle_rad) + yeyeleft(n)*cos(angle_rad);
        xeyeright_new = xeyeright(n)*cos(angle_rad) + yeyeright(n)*sin(angle_rad);
        yeyeright_new = (im_w-xeyeright(n))*sin(angle_rad) +
        yeyeright(n)*cos(angle_rad);
    else
        xeyeleft_new = xeyeleft(n)*cos(-angle_rad) + (im_h-yeyeleft(n))*sin(-
angle_rad);
        yeyeleft_new = xeyeleft(n)*sin(-angle_rad) + yeyeleft(n)*cos(-angle_rad);
        xeyeright_new = xeyeright(n)*cos(-angle_rad) + (im_h-yeyeright(n))*sin(-
angle_rad);
        yeyeright_new = xeyeright(n)*sin(-angle_rad) + yeyeright(n)*cos(-angle_rad);
    end

    bmx = xeyeright_new-xeyeleft_new;

    if strcmp(imageName(6:7),'ba') == 1
        bax = xeyeright_new-xeyeleft_new;
        phi = 0;
    end

    if (strcmp(imageName(6:7),'bb') || strcmp(imageName(6:7),'bc') ||
strcmp(imageName(6:7),'bd') || strcmp(imageName(6:7),'be'))== 1
        phi = +acos(bmx/bax);
    end

    if (strcmp(imageName(6:7),'bf') || strcmp(imageName(6:7),'bg') ||
strcmp(imageName(6:7),'bh') || strcmp(imageName(6:7),'bi'))== 1
        phi = -acos(bmx/bax);
    end
end

```

```

%fprintf('BMX: %d \n', bmx);
%fprintf('View Angle: %d \n', phi*(180/pi));

scale = eyedistance/bax;
im_resized = imresize(im_rotated,scale);

%fprintf('Scale Factor: %d \n', scale);

if phi == 0
    x1 = floor( xeyeleft_new * scale - (norm_width - eyedistance)/2);
end

if phi > 0
    x1 = floor( xeyeright_new * scale + (norm_width - eyedistance)/2 - norm_width
- 20*tan(phi));
end

if phi < 0
    x1 = floor( xeyeleft_new * scale - (norm_width - eyedistance)/2 + 20*tan(-
phi));
end

%%%%%%%%%% Crop Image %%%%%%%%%%%
x1 = floor( xeyeleft_new * scale - (norm_width - eyedistance)/2);
y1 = floor( yeyeleft_new * scale - eyerow );

im_cropped = imcrop(im_resized,[x1 y1 norm_width-1 norm_height-1]);

%%%%%%%%%% Convolute Mask With The Image %%%%%%%%%%%
fid = fopen([dataPath maskFile]);
mask = fread(fid);
fclose(fid);

im_mask = im_cropped;

for i=1:norm_height
    for j=1:norm_width
        if mask((i-1)*norm_width+j) == 0
            im_mask(i,j) = 0;
        end
    end
end

%%%%%%%%%% Histogram Equalize Image %%%%%%%%%%%
im_hist = histeq(im_mask);

%%%%%%%%%% Plot Normalization Steps %%%%%%%%%%%
subplot(2,2,1);imshow(im); title('Original');
subplot(2,2,2);imshow(im_rotated); title('Rotated');
subplot(2,2,3);imshow(im_cropped); title('Cropped');
subplot(2,2,4);imshow(im_mask); title('Masked');
%pause; close;

%%%%%%%%%% Write Normalized Image %%%%%%%%%%%
imwrite(im_hist,[normPath imageName normFormat]);

% Plot
if number ~= 0
    subplot(3,3,plotnumber);imshow(im_hist);title([imageName(6:7) ' '
num2str(phi*(180/pi))]);
    plotnumber = plotnumber+1;
end
end

```

### 11.3 Gabortrain.m

```

function[] = gabortrain()

%%%%%%%%%% Global Definitions %%%%%%%%%%%
dataPath = ['. \DATA\'];
coordFile = ['COORDS3816.TXT'];

normPath = ['. \NORMTOP\'];
normFormat = ['.nrm.pgm'];

```

```

dbPath = ['. \TRAINEDTOP20\'];
dbFormat = ['.mat'];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Collecting Image Data from datafile %%%%%%%%%%%%%%%
[file] = textread([dataPath coordFile], '%s %d %d %d %d %d %d %d %d');
fprintf('Number of files to be trained: %d \n', length(file));

for m=1:size(file) % Iterate through all the images

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Read Training Images %%%%%%%%%%%%%%%
    fprintf('Processing Image ... %d \n', m);

    imageName = char(file(m));
    name = [normPath imageName normFormat];
    im = im2double(imread(name));
    [im_h im_w] = size(im);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Apply Convolution Masks %%%%%%%%%%%%%%%
    n = 1;

    for y=5:20:im_h-5
        for x=5:20:im_w-5
            output(n) = apply_mask0even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask0odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask1even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask1odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask2even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask2odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask3even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask3odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask4even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask4odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask5even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask5odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask6even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask6odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask7even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask7odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask8even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask8odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask9even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask9odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask10even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask10odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask11even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask11odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask12even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask12odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask13even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask13odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask14even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask14odd(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask15even(im, im_w, im_h, x, y); n = n + 1;
            output(n) = apply_mask15odd(im, im_w, im_h, x, y); n = n + 1;
        end
    end

    fprintf('Gabor Image Size n: %d\n', size(output));

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Write Gabor Image to Database %%%%%%%%%%%%%%%
    save([dbPath imageName dbFormat], 'output');

end

```

## 11.4 Gaborrec.m

```

function [candval, candname] = gaborrec(file, number)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
normPath = ['. \NORMTOP\'];
normFormat = ['.nrm.pgm'];

dbPath = ['. \TRAINEDTOP20\'];
dbFormat = ['.mat'];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [file] = textread([dataPath coordFile], '%s %d %d %d %d %d %d %d %d');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read Test Image %%%%%%%%%
imageName = char(file(number));
name = [normPath imageName normFormat];
im = im2double(imread(name));
[im_h im_w] = size(im);

fprintf('Processing Image ... %s ...', imageName);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Apply Convolution Masks %%%%%%%%%
n = 1;

for y=5:20:im_h-5
    for x=5:20:im_w-5
        sample(n) = apply_mask0even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask0odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask1even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask1odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask2even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask2odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask3even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask3odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask4even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask4odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask5even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask5odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask6even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask6odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask7even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask7odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask8even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask8odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask9even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask9odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask10even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask10odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask11even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask11odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask12even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask12odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask13even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask13odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask14even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask14odd(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask15even(im, im_w, im_h, x, y); n = n + 1;
        sample(n) = apply_mask15odd(im, im_w, im_h, x, y); n = n + 1;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Recognition %%%%%%%%%
JETS = 35; % Number of Gabor jets must be changed if number of jets is changed
SCALES = 4;
ROTS = 4;

ROTWIDTH = 2;
SCALEWIDTH = 4*ROTWIDTH;
JETWIDTH = 4*SCALEWIDTH;

```



```

for m=1:size(file) % Do comparison to all the images in the trained database

    if m == number % Exclude test image from recognition
        m = m+1;
    end

    if m > size(file) break; end % Break if test image is last image

    %%%%%%%%%% Load trained Gabor Image from database %%%%%%%%%%%
    load([dbPath char(file(m)) dbFormat], 'output');

    similarity = 0;

    for jet=1:JETS

        sum1 = 0;
        sum2a = 0;
        sum2b = 0;

        for scale=1:SCALES % 4 Scales
            for rot=1:ROTS % 4 Rotations

                index = (jet-1)*JETWIDTH+(scale-1)*SCALEWIDTH+(rot-1)*ROTWIDTH;

                re_sample = sample(index+1);
                im_sample = sample(index+2);

                re_data = output(index+1);
                im_data = output(index+2);

                mag_test = sqrt(re_sample^2+im_sample^2);
                mag_data = sqrt(re_data^2+im_data^2);

                sum1 = sum1 + mag_test*mag_data;
                sum2a = sum2a + mag_test^2;
                sum2b = sum2b + mag_data^2;
            end
        end

        similarity = similarity + sum1/sqrt(sum2a*sum2b); % Similarity measure for
each jet
    end

    similarity = similarity/JETS; % Similarity measure for the whole image
    fprintf('Similarity %d: %d \n', m, similarity);

    %%%%%%%%%% Create similarity vector %%%%%%%%%%%
    candidates(m) = similarity;
end

%%%%%%%%%% Sort similarity vector by descending similarity measure %%%%%%%%%%%
[tempval, tempname] = sort(candidates);

candname = fliplr(tempname); % Training set number representing a file name
candval = fliplr(tempval); % The similarity measure of that file

```

## 11.5 Eigentrain.m

```

function [] = eigentrain(candname2)

%%%%%%%%%% Global Definitions %%%%%%%%%%%
dataPath = ['. \DATA\'];
coordFile = ['COORDS3816.TXT'];
normPath = ['. \NORMTOP\'];
normFormat = ['.nrm.pgm'];
dbFormat = ['.mat'];

fprintf('Training Eigenfaces with PCA...\n');

%%%%%%%%%% Collecting Image Data from datafile %%%%%%%%%%%
[file] = textread([dataPath coordFile], '%s %d %d %d %d %d %d %d %d');
fprintf('Number of files to be trained: %d \n', length(file));

```

```

samples = size(candname2,2);
fprintf('Training Images ... \n');

for m=1:samples % Iterate through all the promising candidates

    %%%%%%%%%% Read Images %%%%%%%%%%
    fprintf('Training Image ... %d \n', m);
    imageName = char(file(candname2(m)));
    im = im2double(imread([normPath imageName normFormat]));
    [im_h im_w] = size(im);

    %%%%%%%%%% Build Training Set Matrix %%%%%%%%%%
    for y=1:im_h
        for x=1:im_w
            trainingSet((y-1)*im_w+x,m)=im(y,x);
        end
    end
end

trainingMean = mean(trainingSet,2); % Average Vector

%%%%%%%%% Subtract Average Vector from Training Set Matrix %%%%%%%%%%
for m=1:samples
    for n=1:(im_h*im_w)
        trainingSet(n,m) = trainingSet(n,m) - trainingMean(n);
    end
end

%%%%%%%%% Build Covariance Matrix %%%%%%%%%%
cov = (trainingSet')*trainingSet;

%%%%%%%%% Calculate the eigenvalues and the eigenvectors %%%%%%%%%%
[eg,ev]=eig(cov);

%%%%%%%%% Sort eigenvectors in order of descending eigenvalue %%%%%%%%%%
ev=real(ev);
ev=diag(ev);
oldev=ev;
[tmp,sortindex]=sort(ev);
sortindex=flipud(sortindex);
ev=ev(sortindex);
eg=eg(:,sortindex);

%A=eg
A=trainingSet*eg;
[A_h A_w] = size(A);

for x=1:A_w
    normen = norm(A(:,x));
    for y=1:A_h
        A(y,x) = A(y,x)/normen;
    end
end

%%%%%%%%% Calculating linear combinations for Training Images %%%%%%%%%%
for m=1:samples

    fprintf('Calculating linear combination for Image ... %d \n', m);

    imageName = char(file(candname2(m)));
    im = im2double(imread([normPath imageName normFormat]));
    [im_h im_w] = size(im);

    for y=1:im_h
        for x=1:im_w
            testSet((y-1)*im_w+x,1)=im(y,x);
        end
    end

    for n=1:(im_h*im_w)

```

```

        testSet(n,1) = testSet(n,1) - trainingMean(n);
    end

    %%%%%%%%%% Build Training Base %%%%%%%%%%
    trainingBase(:,m) = A'*testSet(:,1);
end

%%%%%%%%%% Temporary store trainingBase and trainingMean %%%%%%%%%%
dbName = 'facebase';
save([dataPath dbName dbFormat], 'A', 'trainingBase', 'trainingMean', 'ev');

```

## 11.6 Eigenrec.m

```

function [candval3, candname3] = eigenrec(candname2, number)

%%%%%%%%%% Global Definitions %%%%%%%%%%
dataPath = ['. \DATA\'];
coordFile = ['COORDS3816.TXT'];
normPath = ['. \NORMTOP\'];
normFormat = ['.nrm.pgm'];
dbFormat = ['.mat'];

fprintf('Recognizing Eigenfaces with PCA...\n');

%%%%%%%%%% Read temporary stored trainingBase and trainingMean %%%%%%%%%%
dbName = 'facebase';
load([dataPath dbName dbFormat], 'A', 'trainingBase', 'trainingMean', 'ev');

%%%%%%%%%% Collecting Image Data from datafile %%%%%%%%%%
[file] = textread([dataPath coordFile], '%s %d %d %d %d %d %d %d %d');
fprintf('Number of files in database: %d \n', length(file));

%%%%%%%%%% Read Test Image %%%%%%%%%%
fprintf('Processing Image ... %d \n', number);
imageName = char(file(number));
name = [normPath imageName normFormat];
im = im2double(imread(name));
[im_h im_w] = size(im);

%%%%%%%%%% Calculating linear combination for Test Image %%%%%%%%%%
for y=1:im_h
    for x=1:im_w
        testSet((y-1)*im_w+x,1)=im(y,x);
    end
end

%%%%%%%%%% Build Test Base %%%%%%%%%%
for n=1:(im_h*im_w)
    testSet(n,1) = testSet(n,1) - trainingMean(n);
end

testBase = A'*testSet(:,1);

%%%%%%%%%% Compare Test Image with Promising Trained Candidates %%%%%%%%%%
minValue=10000000;
samples = size(candname2,2);

for m=1:samples % Iterate through all the promising candidates

    %%%%%%%%%%% Mahalanobis Distance %%%%%%%%%%
    Mah = 0;
    for i=1:samples
        Mah = Mah + (trainingBase(i,m)*testBase(i,1)/sqrt(ev(i)));
    end

    Mah = -Mah;

    if Mah < minValue
        minValue = Mah;
        minName = char(file(candname2(m)));
    end
end

```

```

end

if minValue < -1000
    minValue=10000000;

    for p=1:samples % Iterate through all the promising candidates

        %%%%%%%%% L2 Norm %%%%%%%%%
        L2norm = 0;
        for i=1:samples
            L2norm = L2norm + (trainingBase(i,p) - testBase(i,1))^2;
        end

        L2norm = sqrt(L2norm);

        if L2norm < minValue
            minValue = L2norm;
            minName = char(file(candname2(p)));
        end
    end
end

candname3 = minName; % Best Candidate Number
candval3 = minValue; % Best Candidate Value

```

## 11.7 Testhybrid.m

```

function[] = testhybrid()

%%%%%%%% Global Definitions %%%%%%%%%
dataPath = ['.\\DATA\\'];
coordFile = ['COORDS3816.TXT'];

maxCandidate = 0;

%%%%%%%% Collecting Image Data from datafile %%%%%%%%%
[file] = textread([dataPath coordFile], '%s %*d %*d %*d %*d %*d %*d %*d %*d');

for number=1:size(file) % Iterate through the images specified

    imageName = char(file(number));
    fprintf('%d : %s : ', number, imageName);

    [candval, candname] = gaborrec(file, number); % Run Gabor Recognizer

    recName = char(file(candname(1)));
    recVal = candval(1);

    fprintf('Gabor - %s (%d) : ', recName, recVal);

    % Use only the 100 first candidates as promising candidates
    candval2 = candval(1:100);
    candname2 = candname(1:100);

    eigentrain(candname2); % Run PCA Training
    [candval3, candname3] = eigenrec(candname2, number); % Run PCA Recognizer

    fprintf('PCA - %s (%d) : ', candname3, candval3);

    % Verify Correctness of the final choice
    if strcmp(imageName(1:5),candname3(1:5)) == 1
        fprintf('OK\n');
    else fprintf('False\n'); end
end

```

This document was created with Win2PDF available at <http://www.daneprairie.com>.  
The unregistered version of Win2PDF is for evaluation or non-commercial use only.