Face Recognition

Written by Atle Nes (atlen@idi.ntnu.no)
Supervised by Kjetil Bø (ketilb@idi.ntnu.no)

Submitted 22nd of November 2002

Norwegian University of Science and Technology
Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Computer and Information Science
SIF8092 Image Processing Specialization Project

Summary

We present to the reader an introduction to face recognition, and its existing and future applications. Several different existing methods for achieving successful face recognition are presented. The five main categories of face recognition approaches described here are feature based methods, PCA based eigenfaces, LDA based fisherfaces, gabor wavelet based methods, neural networks and hidden markov models.

We also describe an off the shelf simulation environment for image processing operations called Dynamic Imager. Developing your own Dynamic Imager DLL modules is very simple, and they can easily be integrated and used together with several predefined modules. We describe the functionality of some home-made modules and some of the built-in modules. Then we use these modules to construct a working face recognizer consisting of a training network and a recognition network.

Finally we do some testing on the FERET face database to see how good our implementation is. Our focus however is on when it fails, and why it fails. We investigate some common reasons for false identification and try to suggest how this can be improved.

Table of Contents

1 Preface

This is the result of a project assignment in the course SIF8092 Image Processing Specialization Project at the Department of Computer and Information Science, Norwegian University of Science and Technology. The assignment was posted and supervised by Kjetil Bø.

This project has been conducted during 12 weeks in autumn 2002 and is part of my master degree studies in Computer Science. 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. Also thanks to my fellow students Solveig Oldervik Jørgensen and Håkon Johan Thallaug Heuch who have participated in writing some of the subchapters about different face recognition algorithms, but chose other approaches for solving this task.

2 Vocabulary

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

API - Application Programming Interface
A collection of routines or functions which programming applications use to perform different operations. For instance in Microsoft Windows there is an API which is used by applications to manage windows, menus and icons. A single application should have no need to have its own routines to handle this. They can do it by calling a common API.

ASCII - American Standard Code of Information Interchange
Is a standard seven-bit code that was proposed by ANSI in 1963, and finalized in 1968. ASCII was established to achieve compatibility between various types of data processing equipment. The standard ASCII character set consists of 128 decimal numbers ranging from zero through 127 assigned to letters, numbers, punctuation marks, and the most common special characters. The Extended ASCII Character Set also consists of 128 decimal numbers and ranges from 128 through 255 representing additional special, mathematical, graphic, and foreign characters.

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 Π 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.

BZ2
bzip2 is a freely available, lossless data compressor.

C
High-level programming language created by Bell Laboratories. Contains most of modern data- and control structures that are expected to be found in a professional programming language. C is not dependant of being run on any specific machine platform, but was originally built for UNIX.

C++
An object-oriented programming language created by Bjarne Stroustrup. C++ has its roots in C and has become very popular in the way it combines classic C with object-oriented programming.

Canonical Image
Image where size and position of the image is normalized.

Ceetron
Image processing company with main office in Trondheim, Norway. Develops and offers advanced software products and consulting services with 3D visualization and video surveillance. One of their products is the image processing tool Dynamic Imager, used extensively in this project. (http://www.ceetron.com/)

Chess Board Distance
$$d_{chessboard} = \max\left(|x_2 - x_1|, |y_2 - y_1|\right)$$

City Block Distance
$$d_{cityblock} = |x_2 - x_1| + |y_2 - y_1|$$

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.

DLL - Dynamic Linked Library
An implementation method for library functions under the operating systems OS/2 and Windows. The library functions are stored as files with extension DLL, and Windows or OS/2 loads the DLL-file when the running program needs some of the library functions.

DSW – Dev Studio Workspace
ASCII file format used by MS Visual C++ to store information about the workspace.

Dynamic Imager
An image processing visual experimentation environment that allows you to easily set up and test customized sequences of modules containing image processing operations.

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

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.

Heuristic Search
Heuristics, or the art of good guessing comes from the Greek word for discover. Heuristic planning allows machines to recognize promising approaches to problems, break problems into smaller problems, deal with incomplete and ambiguous data and make educated guesses.

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 Π. 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, Π).

IrfanView
Is a freeware graphic viewer for running under Windows.

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/)

Karhunen-Loeve Projection
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.

MDF - Module Declaration Function
Is a mandatory function in a Dynamic Imager DLL module source file. It declares the input and output parameters of the module, as well as user interaction control elements. The function gives Dynamic Imager some basic information about the behaviour of the module.

MIF - Module Implementation Function
Is a mandatory function in a Dynamic Imager DLL module source file. It contains the task algorithm. This function is called under network execution when the module is activated by input or control parameters.

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.

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 Intels 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.

NP – Nondeterministic Polynomial Time Complete Problems
A set of problems that are the hardest problems in the sense that they are the ones most likely not to be polynomial. If you could find a way to solve an NP-complete problem quickly, then you could use that algorithm to solve all NP problems quickly.

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.

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.

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.

TIFF – Tagged Image File Format
TIFF is primarily designed for raster data interchange. It's main strengths are a highly flexible and platform-independent format which is supported by numerous image processing applications.

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.

Viisage Technology
A provider of secure digital identification systems, among them facial recognition technology. (http://www.viisage.com/)

Visual Studio
Visual Studio is a development suite and contains development environments for several different programming languages. It has become an industry standard for C++ and Basic development on Windows operating systems.

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 biometric systems and especially face recognition both for humans and machines. The project assignment text, intention and limitations will also be defined.

3.1 Project background

Biometric systems are systems that identify or verify human beings. They are typically based on some single biometric feature of humans, but several hybrid systems also exist. Bill Gates, founder of Microsoft, has said that “Biometric Systems will be the most important IT innovation of the next several years”. Another statement from the Gartner Group says that “Biometrics is one of the top ten technologies to watch”. [27]

3.1.1 Biometrics

Some of the most important biometric features are based on physical features like hand, finger, face and eye. Hand geometry measures size and distances of the hand. Handprints and fingerprints look at the pattern of ridges and furrows on the skin surface of the palm and at the fingertips. Face recognition is maybe the most natural and well known biometric because humans are so good at it. Iris retina scans of the pattern formed by veins beneath the retinal surface of the eye has been very successful and shows that not even twins have the same iris pattern (Figure 1). Other biometric features are determined by human behaviour like voice, signature and walk. The way humans generate sound from vocal tracts, mouth, nasal cavities and lips is used for voice recognition. Signature recognition looks at the pattern, speed, acceleration and pressure of the pen when writing ones signature. The latest approach in biometric recognition is using human walk as a classifier. [28]

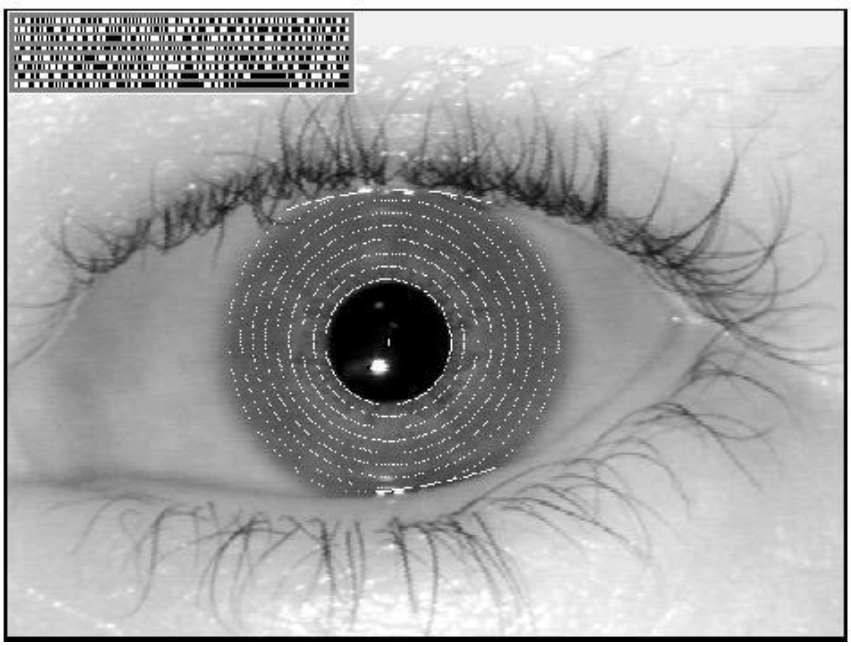


Figure 1: Visualization of iris scan and extracted iris code (upper left corner).

3.1.2 Research

In the 1970 researchers started looking at how human beings recognize faces, and in 1973 the first works on face recognition by machine vision were made by a japanese researcher Kanade. The successes of early works on face recognition from 1973-1990 were often limited, but in 1990 there was a renewed interest in this area because of more powerful computers and an increasing commercial demand. In 1991 the first working algorithm, the so called eigenface method, was invented by Turk and Pentland [9]. In 1995 the first commercial systems were available, and today there are around 20 systems available worldwide. The last couple of years have introduced some new and promising algorithms worth taking a closer look at. Some of them are just extensions of the now well known eigenface algorithm, and others are completely new based on gabor wavelets, neural networks and hidden markov models. We will investigate all these algorithms thoroughly later in a separate chapter.

3.1.3 Human vision

Face recognition is a very important property for human beings who use facial expressions for communication. Accurate face recognition enables primates to identify important individuals in their social structures and avoid aggressive ones. It is actually so important that there are a particular parts of the brain purposed just for face recognition. The first face detection cells were discovered by Gross, Rocha-Miranda and Bender in 1972 [29]. In general faces are quite similar, which makes the recognition very difficult, but humans manage this very well because they get a lot of training. Research shows that new-born children prefer patterns of faces, and already after 1-3 months a child recognizes the faces of its parents. Faces are recognized more easily than other objects. The most important features of the face are eyes, face geometry, hair and mouth. Nose shape is less important. Rough features (low frequency components) are mostly used to decide gender, and more detailed features (high frequency components) are mostly used for identification. The analysis of facial expression can take part in parallel with the face recognition. Tests show that even a sheep, which we consider to be a dumb animal, can distinguish between a sheep from its flock and unacquainted ones. [27] [31]

Figure 2: Human vision face recognition at its superior. Can you find all 13 faces?

3.1.4 Machine vision

When it comes to face recognition by machine vision there are still some major obstacles that reduce the efficiency and success. Frontal recognition is certainly the classical approach of tackling the problem. Modern face recognition has reached an identification rate of greater than 90% for larger databases with well-controlled pose and illumination conditions. View-tolerant algorithms which take into account different poses usually treat the problem in a more sophisticated fashion by taking into consideration some of the underlying physics, geometry and statistics. They are mainly needed in situations where uncooperative or unaware subjects should be authenticated. An area that has received significant attention in recent years is hybrid recognition systems. Different recognition approaches succeed and fail at widely different viewing and illumination conditions. Due to this problem it seems obvious to run various individual recognition classifiers on a problem in parallel, and design an individual ranking system leading to an overall recognition result. Some of the most common obstacles in face recognition are lightning differences, facial hair, cosmetics, facial expression, aging, change of hairstyle and pose variations. Most of the recognition systems available commercially have been demonstrated only with limited datasets, often recorded under restricted conditions. There are however commercially available systems that seem to work quite well. One good example is Visionics FaceIt which reports that a full database search takes less than three seconds with a database size of maximum 30.000 images. [30]

Figure 3: Machine vision face recognition surveillance system at its superior.

3.1.5 Applications

Existing and future applications of face recognition are many. Here follows a list of some of the existing and some imagined applications.

Governmental use:

- **Law enforcement:** Minimizing victim trauma by narrowing mug shot searches. Verifying identity for court records. Comparing school surveillance camera images to an image database of known child molesters.
- **Security/Counterterrorism:** Access control allowing security cleared personnel access. Comparison of surveillance images against an image database of known terrorists and other unwanted people.
- **Immigration:** Rapid progression through customs by using face as a living passport.
- **Election/Legislature:** Verify identity of congressmen or electors prior to vote.
- **Institutions/Prisons:** Tracking of inmates and allowing access to employees.
- **Hospital:** Verify identity of people found unconscious, dead or individuals refusing to identify themselves.
- **Social benefit payments:** Entitlements and benefits authentication. Minimize fraud by verifying identity.

Commercial use:

- **Day Care:** Verify identity of individuals picking up children.
- **Missing Children/Runaways:** Search surveillance images and the internet for missing children and runaways.
- **Premises Access Control:** Access control and personalization of home and office.
- **Internet/E-Commerce:** Biometric key cryptography for encrypting and decrypting secure transactions. Verify identity of purchases on the net and deny/give access to privileged information.
- **Personal Computer:** Login using the face as a living password.
- **Telephone:** Call charging without cash, cards or personal identification numbers (PIN).
- **Banking:** Withdrawing cash from an automated teller machine (ATM) without cards or PIN numbers.
- **Travel:** Ticketless, document-free travelling.
- **Image/Video Indexation:** Search for individuals in databases containing images or video sequences.

3.2 Assignment description

The intention of this specialization project is to make a solid, theoretical platform for the post-graduate thesis the following semester. The work this autumn will in most cases lead to a relevant topic that can be pursued in the spring semester. Often the state of the art in a certain subject is investigated to find out how far technology and research has come, and by doing experiments that prove or disapprove principles and theories. There may also be some construction and software development.

The original assignment text:

Face Recognition

Face recognition is an important task in different types of image understanding. The assignment consists of using image analysis methods combined with artificial intelligence to analyse a face and identify the person from an image database, independent of basis transformations. Face recognition gives many possibilities in among others access control and security.

The implementation is made in MS Visual C++.

One vital matter that is not mentioned in the assignment text is the extensive use of Dynamic Imager. Dynamic Imager is an image processing visual experimentation environment that allows you to easily set up and test customized sequences of modules containing image processing operations. All our implementation in Microsoft Visual C++ will result in DLL-files which are imported and used as modules in Dynamic Imager, together with several predefined ones. We will describe this process more thoroughly later in the implementation chapter.

3.3 Assignment limitations

Detection of faces consists of deciding from a random image if there are any faces in the image and finding their positions and sizes. For a complete system this is just as important as the recognition process to get a good result of an overall recognition system. The output from this detection process is usually canonical images of faces where size and position of the image is normalized.

Because of limited time we decided to focus our entire work on the recognition process itself and not detection of faces. As input we have therefore used face images from FERET, one of the largest face databases. This makes the process a little bit easier because we know that there is exactly one face in every image, and they are quite close to being canonical images. We will describe the FERET database later.

The first phase of the project consisted of gathering information about different face recognition algorithms, and finding out benefits and drawbacks with the different approaches. The second phase consisted of designing, implementing a working solution using gabor wavelets for this purpose. The focus of the second phase has therefore been gabor wavelets, and the other methods are not investigated any further. The third phase was a testing phase for the implementation made. The main mission was to try to see what went wrong in some of the recognition cases.

4 Face recognition algorithms

This chapter will give state of the art information about the most important face recognition algorithms that exist today. We start by looking at the some of the earliest approaches with feature based recognition. Then we take a look at the PCA based eigenface method and the LDA based fisherface method. A quite new biological motivated approach is face recognition using Gabor wavelets. Or maybe neural networks can be used in the recognition process. Finally we take a look at hidden markov models which have had big success in speech recognition.

4.1 Feature based recognition

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 [10] in 1973, and was one of the first approaches to automated face recognition. Later work by Allan L. Yuille [8] describes a method for feature extraction using deformable templates. The feature of interest, an eye for example, is described by a parameterized template. 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. The final parameter values can be used as descriptors for the feature.

4.1.2 Geometric method

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. This implies that the overall geometric properties of features are sufficient for face recognition [11]. The configuration of the features can be described by a vector of numerical data representing the position and size of the main facial features. This information can be supplemented by the shape of the face outline.

4.1.3 Normalization

One of the most critical issues in a geometrical approach is a proper normalization. The image must be normalized in order to be invariant of position, scale and rotation in the image plane. Position invariance is achieved once a point in the image is found with good accuracy which can be used as the origin of coordinates. In [11] they achieve scale invariance by transforming the image so the distance between the eyes and the direction of the axis between the eyes has a predefined size.

To locate the eyes they use a pair of eyes as a template. To cope with scale variations they used the template in 5 different scales (scales 0.7, 0.85, 1, 1.15 and 1.3). They found the position of the eyes by looking for the largest correlation value between the image and each of the scaled templates. The correlation was found using:

Equation 1: $$C_N(x,y) = \frac{< I_T T > - < I_T >< T >}{\sigma(I_T)\sigma(T)}$$

Where I_T is the patch of image I which must be matched to T, <> the average operator, $I_T T$ represents the pixel-by-pixel product, and σ the standard deviation over the area being matched.

When the positions of the eyes are located they first adjust the scale according to the scale of the best fitting template. Then the positions of the left and right eye are refined using a separate template for each eye. Once the eyes have been separately located rotation in the image plane can be corrected by aligning the eye-to-eye axis with the horizon. R. Brunelli and T. Poggio reported good normalization results with this method [11].

4.1.4 Feature extraction

A useful technique for extraction of facial features [10] [11] is that of integral projections. $I(x,y)$ is the image and the vertical projection in the rectangle $[x_1, x_2] \times [y_1, y_2]$ is defined as:

Equation 2: $$V(x) = \sum_{y=y_1}^{y_2} I(x,y)$$

Similarly, the horizontal integral projection is defined as:

Equation 3: $$V(y) = \sum_{x=x_1}^{x_2} I(x,y)$$

Projections can be effective in determining the position of features provided that the window where they act are suitably located to avoid insignificant information. In [11] the images were preprocessed with two gradient operators to get edge images. The two operators produced two different edge maps. One with horizontal edges and one with vertical edges. These edge maps were thresholded to make binary images. The features of the face were then located by running integral projections in the areas where the different features where suspected to be based on the location of the eyes and a priori knowledge of the average human physiology.

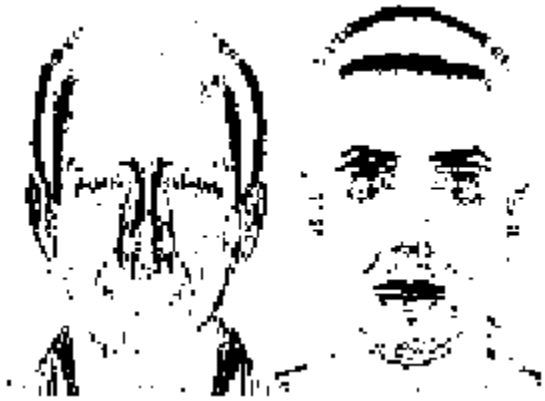


Figure 4: Horizontal and vertical edge maps

The different features were localized by looking for expected variation in the edge projection data. For example the nose was located by looking for peaks of the horizontal projection of the vertical edge map. The mouth was located by looking for a valley in the horizontal projection of the horizontal edge map, due to the dark line between the lips. The eyebrows where localized using a similar strategy, but for detection of the face outline they used a different strategy. Since the face outline is near elliptical they used dynamic programming to follow the outline on a gradient intensity map of an elliptical projection of the face image.

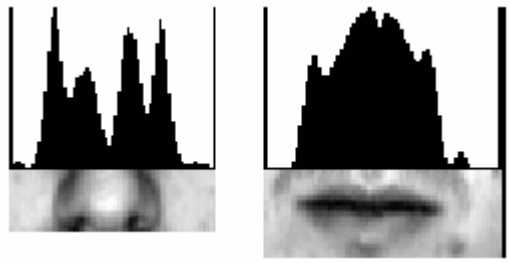


Figure 5: Typical edge projections data

The list of all geometrical features that are extracted automatically in [11] is:

- Eyebrow thickness and vertical position at the eye center position
- A coarse description of the left eyebrows arches
- Nose vertical position and width
- Mouth vertical position, width (upper and lower lips) and height
- Eleven radiuses describing the chin shape
- Face width at nose position
- Face width halfway between nose tip and eyes

Each person is hereby characterized of a 35-dimensional numerical vector, and classification can be done by for example a weighted euclidean distance measure.

4.1.5 Summary

The use of feature vectors seemed 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 therefore sensitive to inaccurate detection of features and to all sorts of disturbance such as facial expressions or varying pose.

4.1.6 Template matching

Another approach to feature based face recognition is template matching. Allan L. Yuille et al. [8] proposed a method for feature extraction based on deformable templates. The templates have a priori knowledge of the expected shape of the different features, and are flexible and adjust to the images to make a best fit by calculating an energy function. The final parameters describing the state of the template can be used to describe the features.

The eye template consists of the following components (Figure 6):

1) A circle of radius r, centered to a point $\vec{x}_c$. The circle corresponds to the boundary between the iris and the white of the eye and is attracted to edges in the image. The interior of the circle is attracted to low values in the image intensity.
2) The bounding contour of the eye is modelled by two parabolas making an elliptic shape. It has a center $\vec{x}_e$, width 2b, maximum height a of the parabola above the center, maximum height c of the parabola below the center, and an angle of orientation θ.
3) Two points corresponding to the centers of the white in the eyes, which are attracted to peaks in the image intensity. These points are labelled by $\vec{x}_e + p_1(\cos\theta, \sin\theta)$ and $\vec{x}_e + p_2(\cos\theta, \sin\theta)$ where $p_1 \geq 0$ and $p_2 \leq 0$.

These components are linked together by three types of forces:

i. Forces which encourage $\vec{x}_c$ and $\vec{x}_e$ to be close together.
ii. Forces which make the width 2b of the eye roughly four times the radius r of the iris.
iii. Forces which encourage the centres of the whites in the eyes to be roughly midway from the centre of the eye to the boundary.

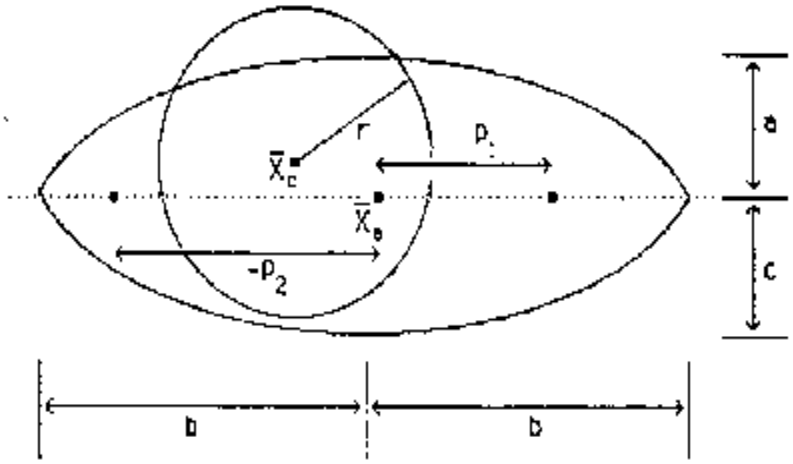


Figure 6: The eye template

There are eleven parameters that are allowed to vary in the eye template. These parameters are adjusted by an energy function seeking the configuration with the lowest possible potential energy. They report problems in [8] with finding good initial coefficients. But after they selected a good set of coefficients the template usually converged to the eye provided that the template was started at it or below it. The configuration of parameters in templates after training can be used in face recognition.

4.2 Eigenfaces

4.2.1 Introduction

Principle Component Analysis (PCA), also known as Karhunen-Loeve 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. Michael Kirby was the first to introduce the idea of the low-dimensional characterization of faces [24]. Then Turk and Pentland worked with eigenspace projection for face recognition [9] and most recently Shree Nayar used eigenspace projection to identify objects using a turntable to view objects at different angles [25].

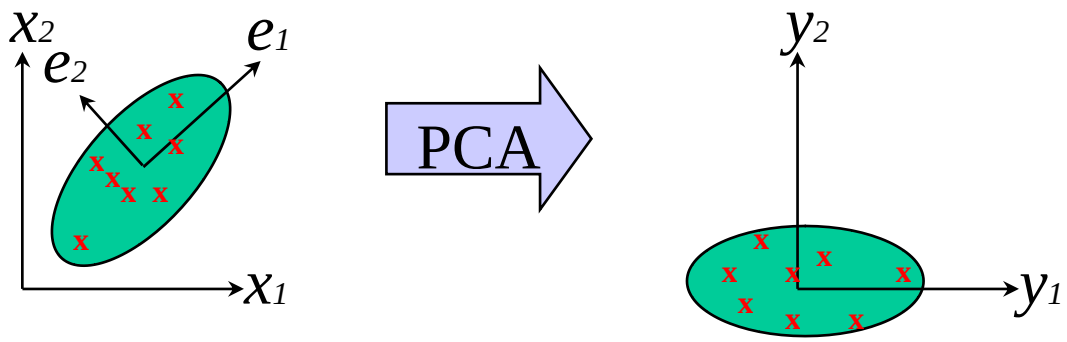


Figure 7: PCA projection means rotating the data so that its primary axes lie along the axes of the coordinate space and move it so that its center of mass lies on the origin.

In this face recognition approach the training images are projected into subspace called eigenspace, consisting of the eigenvectors from the covariance matrix of all the training images. The eigenvectors have a face-like appearance and is therefore also called eigenfaces. Thereafter a test image is projected into the same space and the identification is performed by a distance measure. 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. The extraction of features in this way is done without considering what humans intuitively regard as important features in a face. Unlike Fisher discriminants (next chapter), the eigenface method does not optimize discrimination characteristics, but rather the variance among the images.

4.2.2 The algorithm

Eigenspace is created by calculating the eigenvectors of the covariance matrix of the training images. First pixel data (Equation 4) from the training images $\hat{x}^i$ must be centered. The centered image $\hat{x}^i_{centered}$ is calculated by subtracting the mean of all the training images M from the original image $\hat{x}^i$ (Equation 5). This tells something about how much the training face $\hat{x}^i$ differs from the mean face.

Equation 4: $\hat{x}^i = [x^i_1 \quad x^i_2 \quad \dots \quad x^i_N]^T$

Equation 5: $\hat{x}^{i}_{centered} = \hat{x}_{i} - \frac{1}{M}\sum_{i=1}^{M}\hat{x}^{i}$

These image vectors are then placed column by column in a matrix $\tilde{X}$ (Equation 6) with size NxM (number of pixles N in each image multiplied by number of images M). The data matrix $\tilde{X}$ is then multiplied with its transponate $\tilde{X}^{T}$ to get the covariance matrix $\tilde{K}$ (Equation 7).

Equation 6: $\tilde{X} = [\hat{x}^{1}_{centered} \quad \hat{x}^{2}_{centered} \quad \dots \quad \hat{x}^{N}_{centered}]$

Equation 7: $\tilde{K} = \tilde{X}\tilde{X}^{T}$

Eigenfaces are then the orthogonal eigenvectors of the covariance matrix $\tilde{K}$. If the covariance matrix $\tilde{K}$ is not too big we just calculate eigenvalues λ and eigenvectors $\hat{e}$ (Equation 8). The eigenvectors are then ordered descending by looking at the associated eigenvalues. Only eigenvectors with eigenvalues different from zero are kept. The eigenspace matrix $\tilde{E}$ consists of all these eigenvectors placed column by column (Equation 10).

Equation 8: $\tilde{K}\tilde{E} = \hat{\lambda}\tilde{E}$

Equation 9: $\hat{\lambda} = [\lambda_{1} \quad \lambda_{2} \quad \dots \quad \lambda_{M}]^{T}$

Equation 10: $\tilde{E} = [\hat{e}_{1} \quad \hat{e}_{2} \quad \dots \quad \hat{e}_{M}]$

The eigenvector with the highest eigenvalue will find the greatest variance in the images, and the eigenvector with the lowest eigenvalue finds the least variance among the images. Usually the covariance matrix $\tilde{K}$ is too large to work with and it is therefore common to use a simplified calculation approach. Since the number of training images usually is less than the number of pixels in the image there will be only M-1 and not N interesting eigenvectors.

4.2.3 The recognition process

The recognition is performed by projecting a test image into the 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 chess board distance and euclidean distance. Euclidean distance is the common method. 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.

4.2.4 Summary

In [15] the performance of the eigenspace method is tested. The results show that the method is fairly good for head-on images that are taken at the same time, but performance decreases rapidly as the time between training and test images increases. The test also indicates that the method has some problems with different light conditions and rotation of image subject.

4.2.5 Modular and View-Based Eigenspace

Modular eigenspace, also called eigenfeatures, is an extended technique that uses not only the face as a whole like the standard eigenfaces method, but also smaller features like eyes, nose, mouth etc. The computation is the same as with eigenfaces method, but will end up with eigeneyes, eigennoses and eigenmouths instead. View-Based methods try to take into account the different pose variations, and not only the frontal face image. More about these extensions can be read in [26].

4.2.6 Eigen Light-Field

A variant of eigenfaces is proposed by Ralph Gross et. al in [12]. They calculate the light-fields of faces and use the light-fields instead of raw images to calculate eigenspace. The light-field is also called the plenoptic function [13] and is a function which specifies the radiance of light in free space. It is generally a 5D function of position (3D) and orientation (2D) , but in the case where there is no scattering or absorption of light through the air, the light field is only a 4D function. It is a 2D function of position defined over a 2D surface, and a 2D function of direction [14].

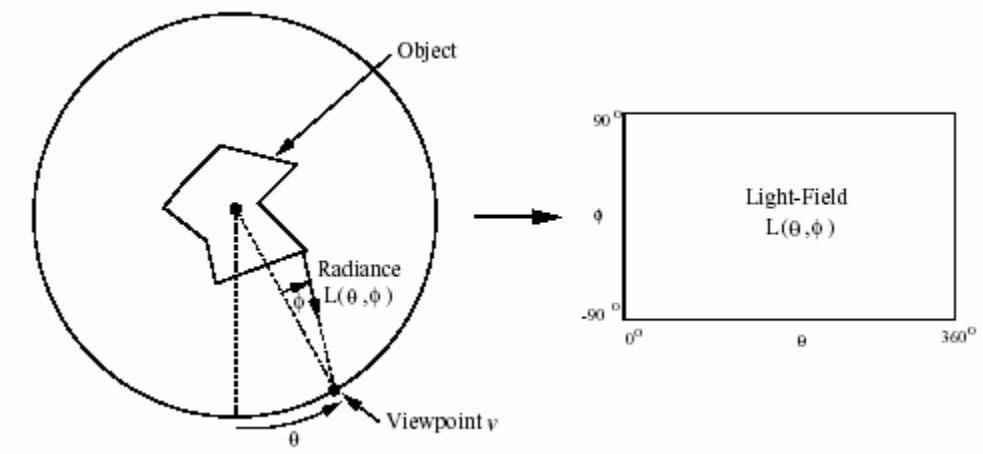


Figure 8: An illustration of the 2D light-field of a 2D object.

The picture above illustrates the light field function $L(\theta,\phi)$. The object is placed within a circle and the viewpoint *v* is placed on the circle with an angle θ measured from the bottom position. The angle that the light-ray has to the radius is denoted by ϕ. For each pair of angles (θ,ϕ) the amount of radiation reaching the viewpoint is denoted $L(\theta,\phi)$. Calculation of the light-field function is described in [14]. In the paper of [12] they achieve good results with head-on images, but it seems like the biggest advantage with the method is the ability to recognize face images in arbitrary poses.

4.3 Fisherfaces

4.3.1 Introduction

Two sets of points, green and blue, in two-dimensional space are projected onto a single line. Depending on the direction of the line, the points can either be mixed together or be separated (Figure 9).

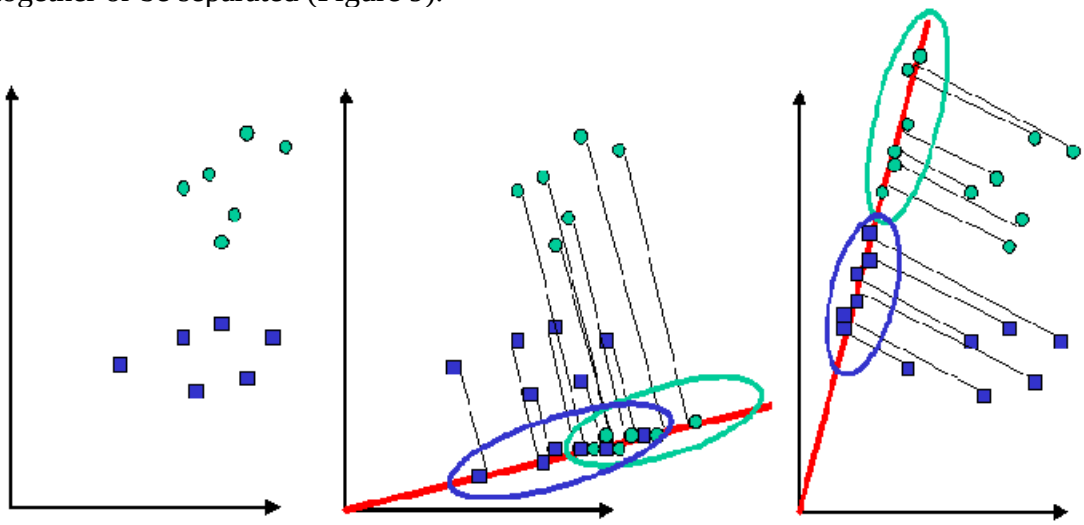


Figure 9: (a) Points in two-dimensional space (b) poor separation (c) good separation

R. A. Fisher developed Fisher's Linear Discriminant (FLD) [3] 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 [2]. Also, Belhumeur, Hespanha and Kriegman used FLD to identify faces, by training and testing with several faces under different lighting [1].

Fisher discriminant analysis, also called Linear Discriminant Analysis (LDA) 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. 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. [4][5][6][7]

4.3.2 Fisherfaces algorithm (original method)

As with eigenspace projection, training images are projected into a subspace. The test images are projected into the same subspace and identified using a similarity measure. The difference compared to the eigenspace projection is how the subspace is calculated. A typical LDA uses Scatter Matrix Analysis.

1. Calculate the within class scatter matrix S_W. This matrix measures the amount of scatter between items in the same class. The scatter matrix S_i belonging to class C_i is calculated as the sum over the covariance matrices of the centered images in the class (Equation 11), where m_i is the mean of the images in the class C_i. The within class scatter matrix S_W is the sum over all scatter matrices (Equation 12).

Equation 11: $S_i = \sum_{x \in C_i} (x - m_i)(x - m_i)^T$

Equation 12: $S_W = \sum_{i=1}^{C} S_i$

The mean of the images in the class m_i is defined as the sum of all items x in the class C_i divided by the number of items in the class C_i (Equation 13).

Equation 13: $m_i = \frac{1}{n_i} \sum_{x \in C_i} x$

2. Calculate the between class scatter matrix S_B. This matrix measures the amount of scatter between classes. It is calculated as the sum of the covariance matrices of the difference between the total mean and the mean of each class (Equation 14), where n_i is the number of.images in the class, m_i is the mean of the images in the class C_i and m is the mean of all the images.

Equation 14: $S_B = \sum_{i=1}^{C} n_i (m_i - m)(m_i - m)^T$

The mean of all the images m is defined as the sum of all items divided by the total number of items (Equation 15).

Equation 15: $m = \frac{1}{N} \sum_{k=1}^{N} x_k$

3. Solve the generalized eigenvectors V and eigenvalues Λ problem of the within class and between class scatter matrices (Equation 16).

Equation 16: $S_B V = \Lambda S_W V$

4. Sort the eigenvectors by their associated eigenvalues from high to low and keep the first M-1 non-zero eigenvectors. These eigenvectors form the Fisher basis vectors, and are also called fisherfaces.

5. Project all the original images onto the Fisher basis vectors by calculating the dot product of the image with each of the Fisher basis vectors. The original images are projected onto this line because these are the points that the line has been created to discriminate, not the centered images.

4.3.3 Fisherfaces algorithm (orthonormal basis method)

To problems arise when using Fisher discriminants. First the matrices needed for computation are very large, causing slow computation time and possible problems with numeric precision. Second, since there are fewer training images than pixels, the data matrix is rank deficient. It is possible to solve the eigenvectors and eigenvalues of a rank deficient matrix by using a generalized singular value decomposition, but an easier solution exists. This solution is to project the data matrix of training images into an orthonormal basis of size PxP, where P is the number of training images. This projection produces a data matrix of full rank that is much smaller and therefore decreases computation time. The projection also preserves information so that the final outcome of Fisher discriminants is not affected.

1. Center all the images in each class (Equation 17) and center the class means (Equation 18).

Equation 17: $\hat{x} = x - m_i$

Equation 18: $\hat{m}_i = m_i - m$

2. Create a data matrix where all the images are combined side by side into one data matrix. Find an orthonormal basis for this data matrix by using a QR Orthogonal-triangular decomposition or by calculating the full set of eigenvectors of the covariance matrix of the training data. Let the orthonormal basis be W.

3. Project all centered images into the orthonormal basis. Create vectors that are the dot product of the image and the vectors in the orthonormal basis (Equation 19). Project the centered means into the orthonormal basis. Create vectors that are the dot product of the centered mean and the vectors in the orthonormal basis (Equation 20).

Equation 19: $\tilde{x} = W^T \hat{x}$

Equation 20: $\tilde{m}_i = W^T \hat{m}_i$

4. Calculate the within class scatter matrix (Equation 21) and the between class scatter matrix (Equation 22).

Equation 21: $S_w = \sum_{i=1}^{C} \sum_{x \in C_i} \tilde{x}\tilde{x}^T$

Equation 22: $S_B = \sum_{i=1}^{C} n_i \tilde{m}_i \tilde{m}_i^T$

5. Solve the generalized eigenvectors V and eigenvalue Λ problem of the within class and between class scatter matrices (Equation 23).

Equation 23: $S_B V = \Lambda S_W V$

6. Sort the eigenvectors by their associated eigenvalues from high to low and keep the first M-1 non-zero eigenvectors. These eigenvectors form the Fisher basis vectors, and are also called fisherfaces.

7. Project all the rotated original images onto the Fisher basis vectors by calculating the dot product of the image with each of the Fisher basis vectors. First project the original images into the orthonormal basis, and then project these projected images onto Fisher basis vectors. The original rotated images are projected onto this line because these are the points that the line has been created to discriminate, not the centered images.

4.3.4 Summary

The fisherfaces method is insensitive to large variations in lightning direction and facial expression. Compared to eigenfaces this algorithm has similar computational requirements, but show lower error rates. It general this 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.

4.4 Wavelets

4.4.1 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. [16]

In the same way as Fourier analysis, wavelets are derived from a basis function $\Phi(x)$ called the mother function or analyzing wavelet. The simplest mother function is the Haar which is a simple step function. Some other well known mother functions are Meyer and Daubechies.

4.4.2 Gabor wavelets

The representation of local features is based on the Gabor wavelet transform. Gabor wavelets are biological motivated convolution kernels in the shape of plane waves restricted by a Gaussian envelope function. The set of convolution coefficients for kernels of different orientations and frequencies at one image pixel is called a jet (Figure 10).

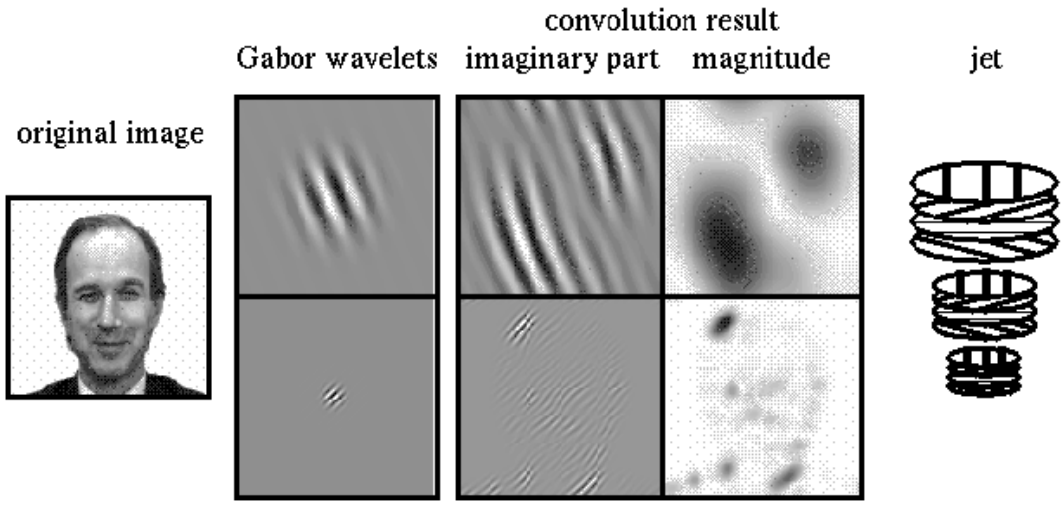


Figure 10: (a) Original image (b) Gabor wavelets (c) Convolution result (d) jet

A jet $J_j(x)$ describes a small patch of grey values in an image $I(x)$ around a given pixel x (Equation 24) with a family of Gabor kernels $\Phi_j(x)$ (Equation 25) in the shape of plane waves with wave vector k_j (Equation 26) restricted by a Gaussian envelope function.

Equation 24: $J_j(x) = \int I(x)\Phi_j(x - x')d^2x'$

Equation 25: $\Phi_j(x) = \frac{k_j^2}{\sigma^2}\exp\left(-\frac{k_j^2 x^2}{2\sigma^2}\right)\left[\exp(ik_j x) - \exp\left(-\frac{\sigma^2}{2}\right)\right]$

Equation 26: $k_j = \begin{pmatrix} k_{jx} \\ k_{jy} \end{pmatrix} = \begin{pmatrix} k_v \cos\varphi_u \\ k_v \sin\varphi_u \end{pmatrix}$, $k_v = 2^{-\frac{v+2}{2}\Pi}$, $\varphi_u = \frac{u\pi}{8}$

40 different coefficients, index j = u + 8v, are computed from 5 different frequencies, index v = 0,…,4, and 8 different orientations, index u = 0,…,7. This sampling evenly covers a band in frequency space. The set J is defined as the set created from these 40 complex coefficients obtained for one image point. Because the Gabor kernels are DC-free a jet can be written as $J_j(x) = a_j \exp(i\Phi_j)$ with amplitudes $a_j(x)$ which slowly vary with position, and phases $\Phi_j(x)$ which rotate with a rate set by the wave vector k_j of the kernels.

Comparing jets can be 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 (Equation 27) or compensate for its variation explicitly.

Equation 27: $S_a(J, J') = \frac{\sum_j a_j a'_j}{\sqrt{\sum_j a_j^2 a'^2_j}}$

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. [17]

4.4.3 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, the tip of the nose and other contour points (Figure 11).

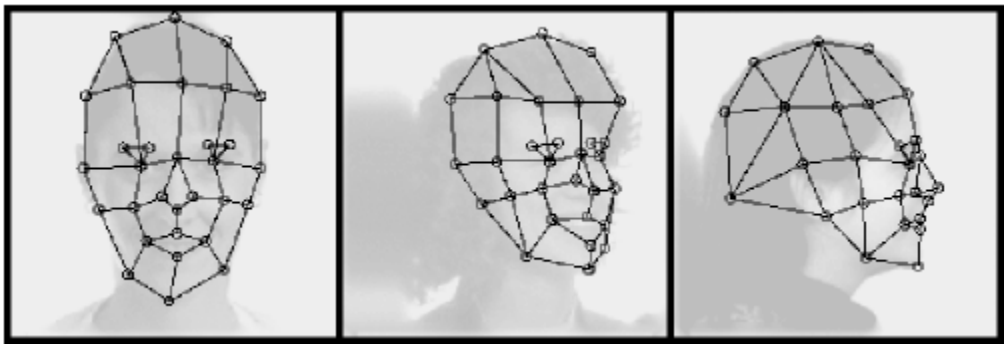


Figure 11: Graphs for faces in different views

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. 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 (Figure 12b). It is crucial that the individual graphs all have the same structure and that the nodes refer to the same fiducial points.

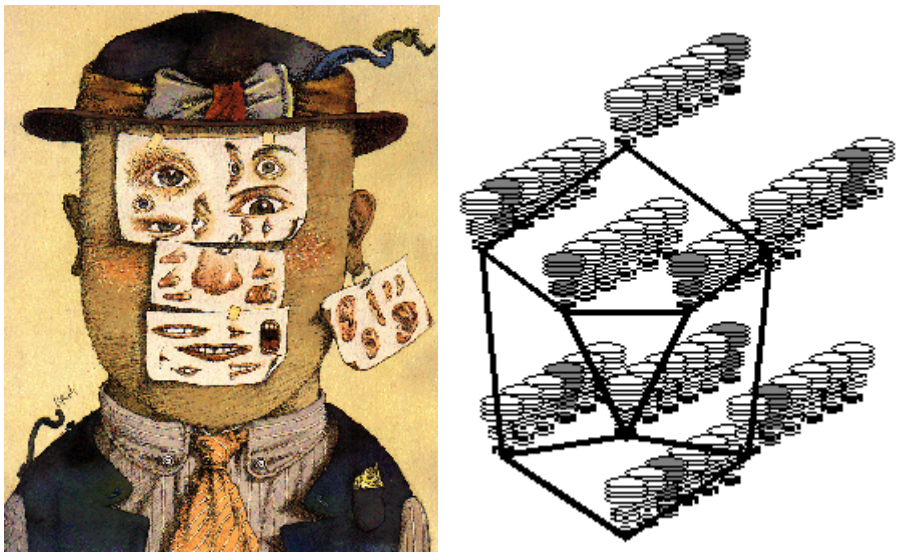


Figure 12: Face Bunch Graph from (a) an artistic point of view, and (b) a scientific point of view.

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.

4.4.4 Summary

Gabor wavelets are chosen for their robustness as a data format and for their biological relevance. Since they 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. Gabor wavelets have similar shape as the respective cells found in the visual cortex of vertebrate animals. 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 potentially represent 6^9 or about 10 million different faces.

4.5 Neural networks

4.5.1 Introduction

Neural networks have been used as an approach in many pattern recognition tasks. Principal component analysis (PCA) is often applied to the neural network approach to give the network only the relevant information, represented by coefficients, instead of having to use the whole image as input. The neural network computes an output according to a given input. Neural networks are used to recognize the pattern, or in this case faces, by learning a correct classification of the coefficients calculated by some sort of PCA of the original face image.

Different methods of feature extraction have been experimented with in combination with neural networks. For face recognition we need a feature vector from a given image. One study used the KLT, also known as eigenfaces to extract relevant information as input to the network [18]. Another has used fourier descriptors [19].

When using PCA the complexity of the system dramatically decreases. The recognition is faster due to the reduced size of the input vector, and as a result, less training time is required [18].

4.5.2 Training

In most cases of neural network face recognition a feed forward network using the backpropagation algorithm is chosen. During training the weights are tuned as a result of the error between the expected output and the actual estimated output. The learning algorithm is called backpropagation. The backpropagation method is a method to find the weights when using a multilayered feed forward network [20]. Given a set of input patterns with associated known outputs (also known as supervised learning) the objective is to train the network to estimate the functional relationship between the inputs and outputs. The squared sum of errors between the given output, and the actual output from the network during training should be as small as possible (Equation 28). If the difference between the predefined output and the actual output is too big the network weights are updated.

Equation 28: $E = 1/2 \sum_{p=1}^{n} \sum_{k=1}^{O} (y_{pk} - \ddot{y}_{pk})^2$

Backpropagation algorithm:

1. Initialize the weights w_{ij} to small random values.
2. Choose a pattern p represented by a vector $[X_1, ...X_n]$ and propagate it forward.
3. Compute the output errors
4. Compute the hidden layer errors
5. Compute the squared error function and find the changes in w_{ij}, Δw_{ij}. Update the weights.
6. Repeat the steps for each pattern the network is supposed to learn

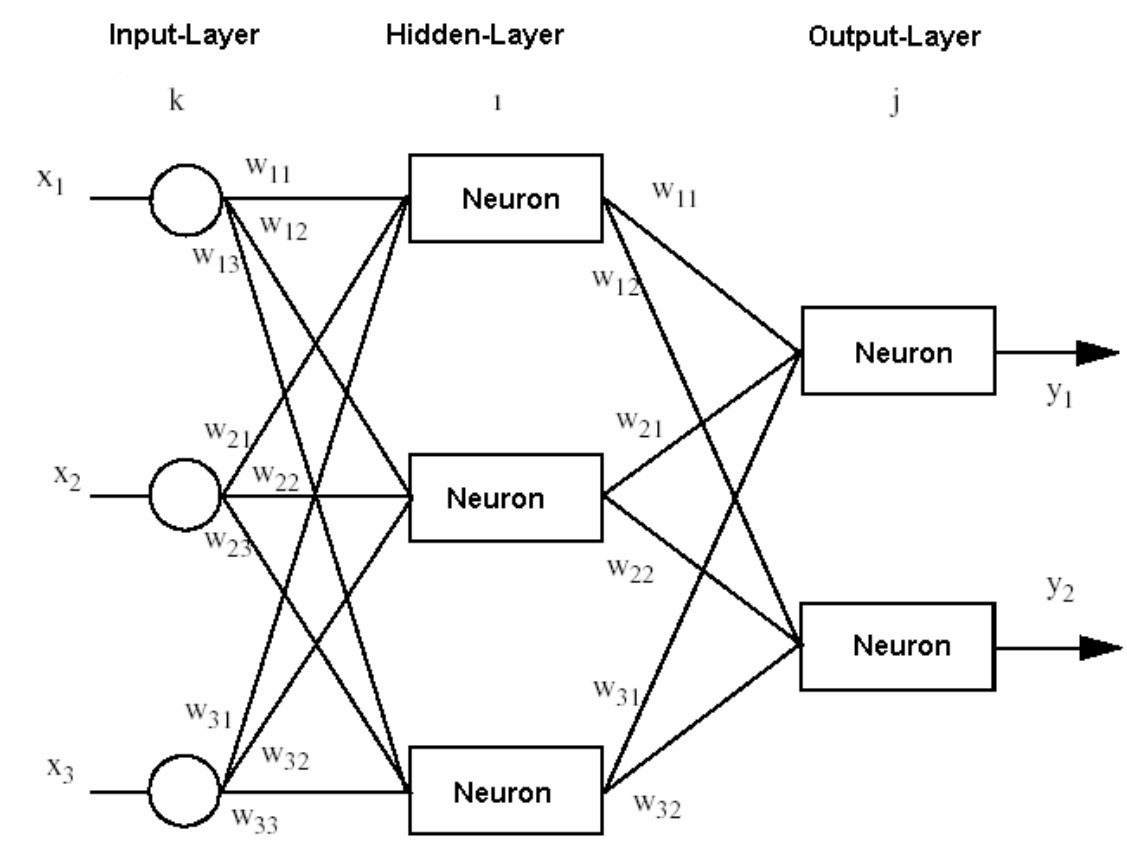


Figure 13: Feed forward neural network

The figure above shows a typical feed forward neural network. We have an input layer consisting of input nodes and an output layer consisting of output nodes. The input nodes are connected to the output nodes via one or more hidden layers (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.

4.5.3 Recognition

When the network has been trained by a learning algorithm, it should be ready to identify face pictures given to it. The unknown test image is transformed to a vector given as input to the network in the same manner as with the training images. If the network is trained properly it should give the right output. Test images must absolutely not be taken from the training set. This is a dirty trick of getting good recognition results, but does not prove anything.

Some neural networks approaches using eigenfaces and fourier descriptors show very good results on limited datasets. One neural network using fourier descriptors was tested with 200 images, 20 for each face. The images were containing faces with different scaling and orientations. Results show that the network can recognize 98% from these patterns correctly. [19]

4.5.4 Summary

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 (Figure 13), 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.

4.6 Hidden markov models

4.6.1 Introduction

Hidden Markov Models (HMM) are a set of statistical models used to characterize the statistical properties of a signal. HMM generally work on sequences of coherent 1D signals (feature vectors), while an image usually is represented by a simple 2D matrix.

In the case of using a 1D HMM the recognition process is based on a frontal face view where the facial regions like hair, forehead, eyes, nose and mouth come in a natural order from top to bottom. Each of these regions is assigned to a state in a left to right 1D continuous HMM (Figure 14).

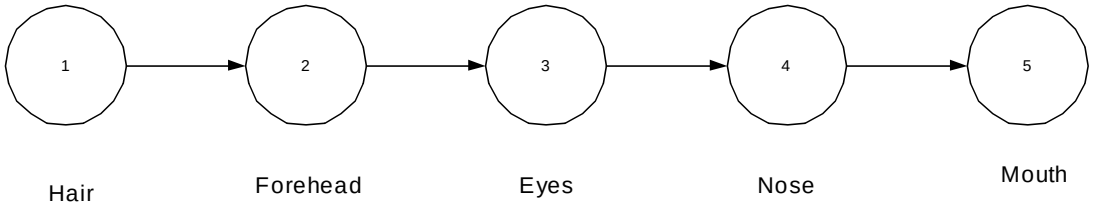


Figure 14: Left to right HMM for face recognition

The image is then vertically scanned from top to bottom (Figure 15) forming a 1D observation sequence. The observation sequence is composed of vectors that represent the consecutive horizontal strips, where each vector contains the pixel values from the associated strip. The vertical overlap is permitted up to one pixel less than strip width [32].

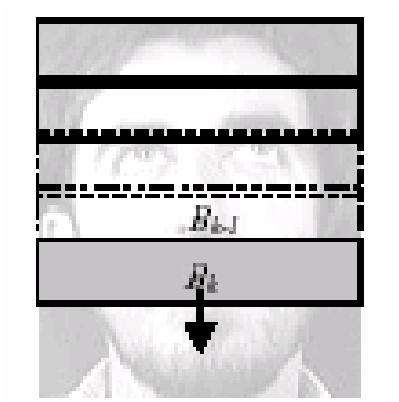


Figure 15: Image scanning face with 1D HMM

2D Pseudo HMM are designed to deal with 2D signals and has just been introduced for face recognition applications. It is composed of a 1D HMM whose states are called superstates. Each of these superstates contains a 1D HMM arranged in the transverse dimension, top to bottom (Figure 16).

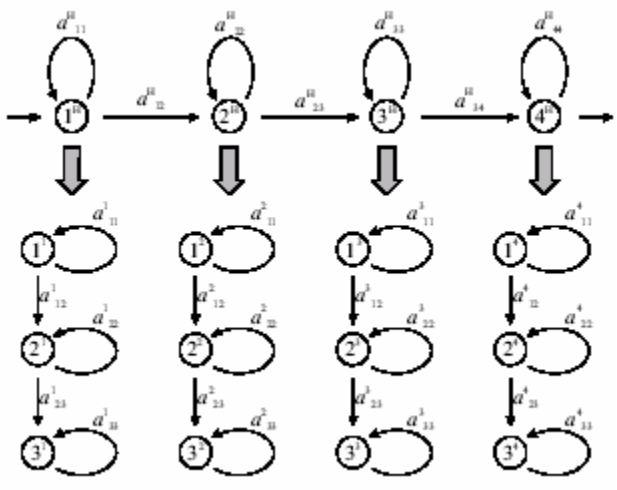


Figure 16: Model topology for 2D Pseudo HMM

The images are divided into blocks. The image is scanned left to right, and top to bottom (Figure 17) to generate an observation sequence.

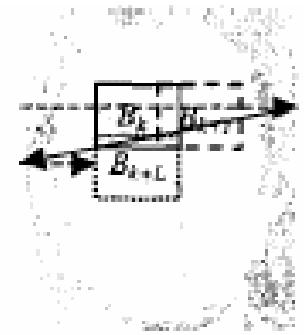


Figure 17: Image scanning face with 2D Pseudo HMM

True 2D HMM handle the picture in full 2D manner without converting the problem into 1D form (Figure 18).

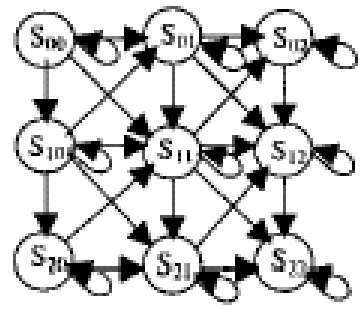


Figure 18: Model topology for 2D HMM

The image is scanned in a 2D manner into 8x8 pixel blocks scanning the image from left to right and from top to bottom (Figure 19). Blocks in diagonal and anti diagonal neighbourhood are assumed independent.

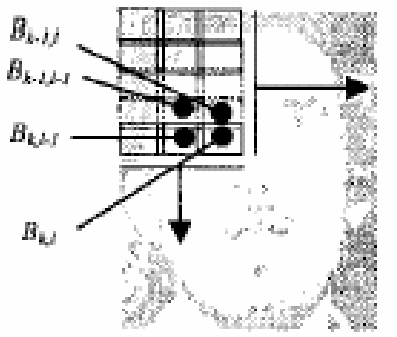


Figure 19: Image scanning face with 2D HMM

In all cases, the observation vectors consist of all the pixel values from each of the strips or blocks scanned from the image. They are represented either as raw pixel values or as PCA values. The use of observation vectors represented by PCA values reduces the size of the vector and also decreases the complexity of the recognition system. In [21] the observation vectors consist of Karhunen-Loeve Transform (KLT) coefficients [22]. In [23] the 2D DCT (Discrete Cosine Transform) coefficients extracted from each block image were used to obtain a set of observation vectors.

4.6.2 Elements of the hidden markov model

N: The number of states in the Hidden Markov Model. If S is the set of states, then can be written as $S = \{S_1, S_2, ...S_n\}$. The state of the model at time t is given by $q_t \in S, 1 \le t \le T$, where T is the length of the observation sequence (number of strips/blocks).

$\prod$: The initial state probability, for instance $\pi = \{\pi i\}$, where $\pi i = P[q_1 = S_i], 1 \le i \le N$

A: The state transition probability matrix, $A = \{a_{ij}\}$, where $a_{ij} = P[q_t = S_j \mid q_{t-1} = S_i]$, $1 \le i,j \le N$, with the constraint, $0 \le a_{ij} \le 1$, and $\sum_{j=1}^{N} a_{ij} = 1$, $1 \le i \le N$.

B: The observation probability matrix. It contains either discrete probability distributions or continuous probability density function of the observations given the state. B = {bj(k)}, where $b_j(k) = P[O_t = v_k \mid q_t = S_j]$, $1 \le j \le N$, $1 \le k \le M$.

O_t is the observation symbol at time t, and v_k is an observation symbol. Short hand, HMM is defined as the triplet λ = **(A , B , $\prod$)**.

4.6.3 Training

Each individual in a database is represented by a HMM face model. A set of images from each person is used to train the corresponding HMM. When the blocks are extracted from each image in the training set, the observation vectors are obtained and used to train each of the models.

First $\lambda = (A, B, \prod)$ is initialized. Then the training data is uniformly segmented from top to bottom, and the observation vectors associated with each state are used to obtain initial estimates of the observation probability matrix B. The initial values of A and $\prod$ are decided given to the left to right structure of the face model.

The training continues using Viterbi segmentation. The HMM is initializes when the Viterbi segmentation likelihood is smaller than a given threshold. The final parameters of the HMM are obtained using the Baum-Welch recursive procedure [21]. The training scheme is shown in Figure 20.

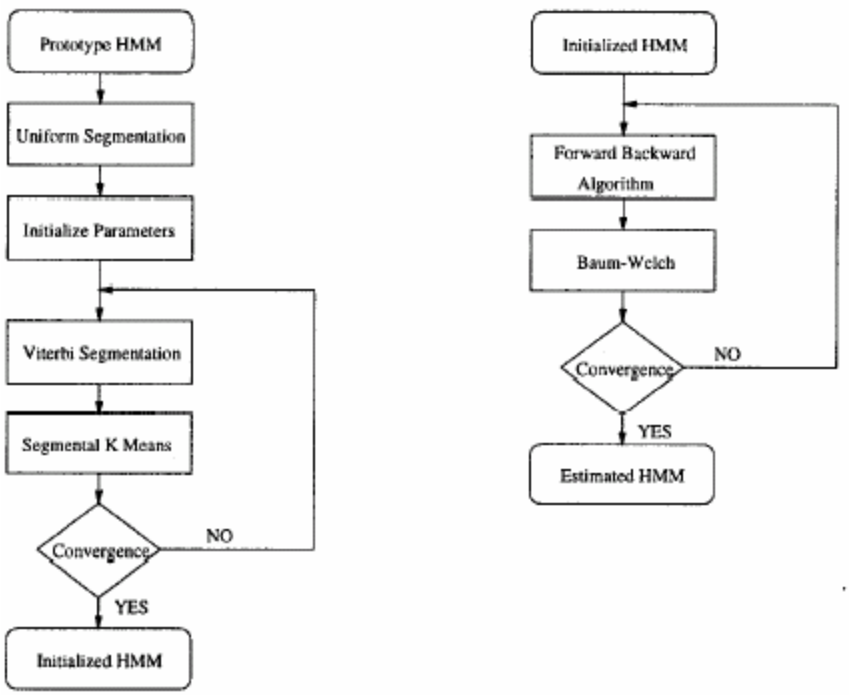


Figure 20: Training scheme

4.6.4 Recognition

Recognition is carried out by matching the test image against each of the trained models. To do this the image is converted to an observation sequence and then the model likelihood $P(O \mid \lambda)$ is computed for each λ.

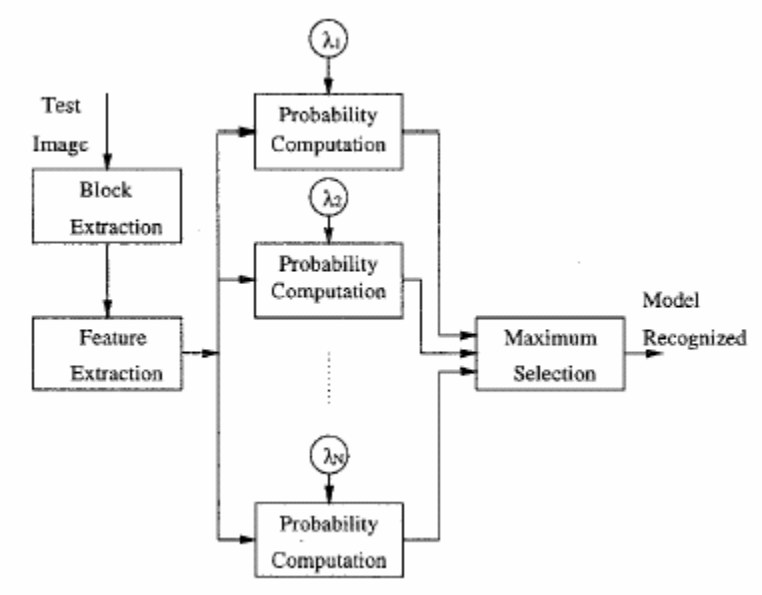


Figure 21: Recognition scheme

Recognition could be done by using a simple Viterbi recognizer as described in [33]. The Viterbi recognizer could be modified to solve the 2D Pseudo HMM and the 2D HMM search problem as well as the 1D HMM case. A collection of HMMs each representing a different subject is matched against the test image and the highest match is selected. The model with the highest likelihood is selected and this model reveals the identity of the unknown face.

One HMM recognition system was tested on the Olivetti Research Database with 400 images, 40 individuals, 10 faces per individual. Half of the images were used in training, the other half in testing. The images were showing different facial expressions, hair styles and eye wear. The system achieved a recognition rate of 87% using 10 KLT features or 39 2D DCT features. The performance compared to the eigenface method used on the same database, which achieved a recognition rate of 73%, shows some of the power of HMM [22].

The 2D Pseudo HMM has achieved a recognition accuracy rate at more than 90% [34]. Its complexity is larger than that of the 1D HMM due to the large number of states in the model, which increases the complexity of the recognition task.

4.6.5 Summary

The advantage of the HMM based approach is its ability to handle variations in scale, which is a challenging problem for a face recognition system. It also has a computational efficiency compared to other approaches. Compared to other face recognition methods the training sets are relative small [22]. We see that compared to the eigenface method based on the same database, HMM gives a better recognition rate.

5 Face recognition implementation

This chapter contains all information about how to implement a face recognition system. First a short introduction to the tools we have used will be presented. Later we describe an implementation of a face recognition system which can be simulated and tested using the Dynamic Imager environment. A detailed description of all the system modules we have been using and their functionality is given, as well as both some simple and some more complex module network architectures.

5.1 Module development process

This subchapter contains a simple description of the two tools we have used, Dynamic Imager and MS Visual C++ 6.0, and how to program a Dynamic Imager DLL module.

5.1.1 What is Dynamic Imager?

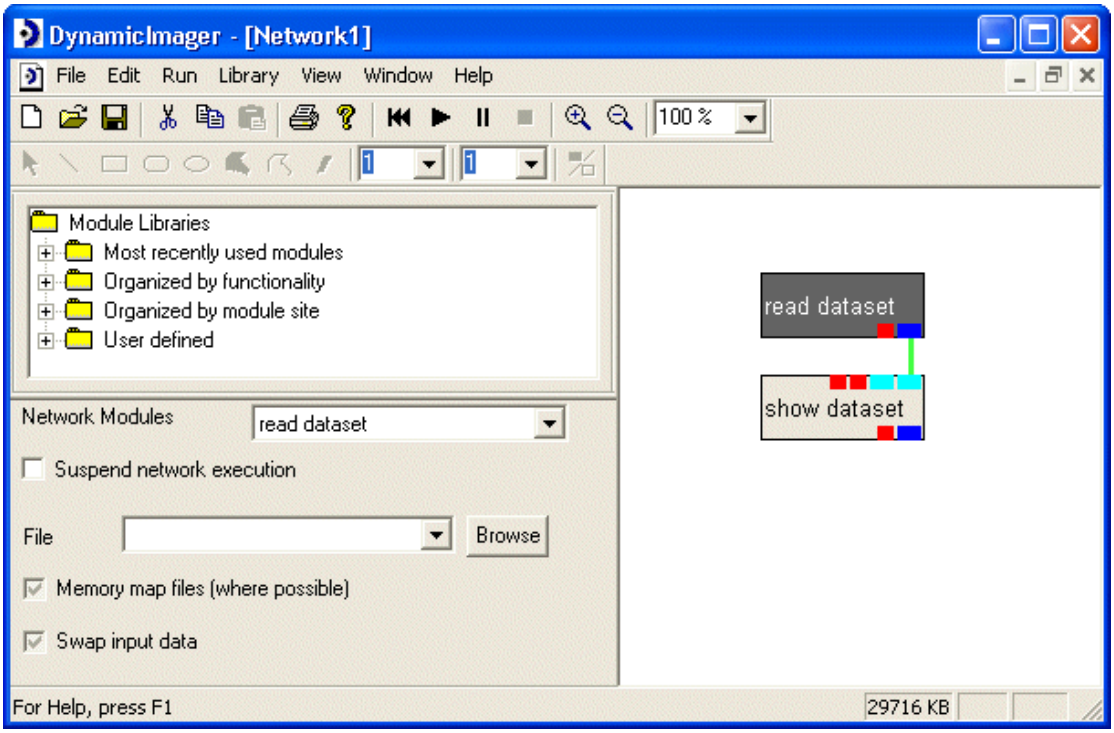


Figure 22: Dynamic Imager graphical user interface and a simple network example

Dynamic Imager is an image processing visual experimentation environment invented by the Norwegian company Ceetron in Trondheim. The program makes it easy to set up and test customized sequences of image processing operations. Dynamic Imager is targeted at users interested in experimenting with image processing, as well as at researchers needing a fast experimentation environment to analyze their data. Dynamic Imager is also targeted at the image processing expert who wants to write image processing algorithms and see them perform without having to write a lot of additional code. Dynamic Imager comes with an API that allows anyone to implement image processing modules and integrate them with the system, as well as using the predefined ones offered by the system. Dynamic Imager modules are linked together

to form networks that can perform both simple and complex tasks. There are many built-in modules which can do standard tasks like enable read and write to many common file formats, show data to screen or perform image transformations like scaling and rotation.

5.1.2 What is Visual C++ 6.0?

User defined modules are created as MS Windows DLL files. All our programming of DLL files has been done with MS Visual C++ 6.0.

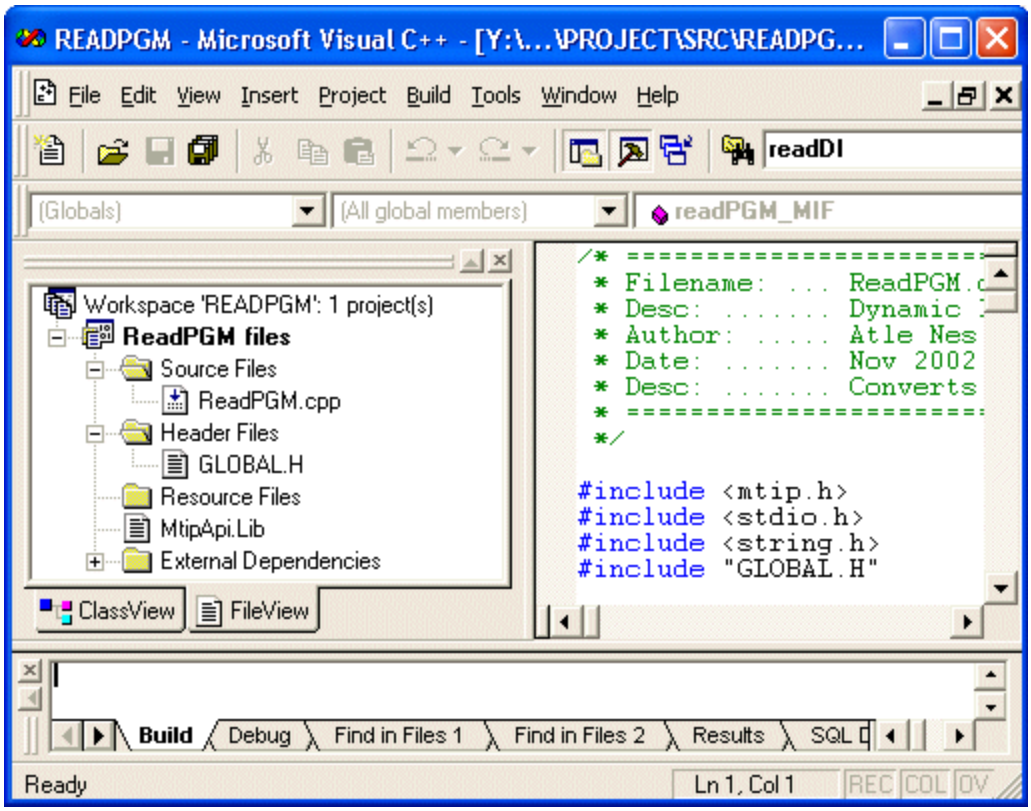


Figure 23: MS Visual C++ 6.0 graphical user interface and programming environment

MS Visual C++ is a member of the MS Visual Studio 6.0 development suite and has become the industry standard for C++ development on Windows operating systems. It contains a complete integrated development environment including debugging support with a look and feel that should be familiar to users of other Microsoft products. Sometimes the extensive functionality can make it difficult to locate the wished menu item or setting.

5.1.3 How do I create a module?

New modules are made by creating a new project of type Win32 Dynamic-Link Library with MS Visual C++, or by copying an old well working workspace. The file library file mtipapi.lib contains the Dynamic Imager API functions, and must be included in the project workspace.

```
#include <mtip.h>

int myMDF(HMTIPMOD hMod)
{
      // Module definition
}

int myMIF(HMTIPMOD hMod, . . . )
{
      // Module algorithm
}

REGISTER_MTIP_MODULE("MyModule", MyMDF, MyMIF, 0, "myhelp.hlp, 1);
```

Table 1: Skeleton for a Dynamic Imager DLL file

A Dynamic Imager DLL module source file must include two mandatory functions. One is the module declaration function (MDF) which declares the input and output parameters of the module, as well as user interaction control elements. This function gives Dynamic Imager some basic information about the behaviour of the module, and how to connect it to other modules in the network. The second mandatory function is the module implementation function (MIF) which contains the task algorithm. This function is called under network execution when the module is activated by input or control parameters. REGISTER_MTIP_MODULE is used to register the module in the Dynamic Imager environment. It contains the name of the module, the MDF function, the MIF function and an optional help file.

Inside the MIF the programmer can use all the functionality provided by the Dynamic Imager API. One of the nice features is to make the module report back to the user. This can be done using the API functions mmsg_Report() or mmsg_ReportError() which will present a dialogue box with information, or mm_SetModuleProgress() which updates the status bar. The first two will suspend execution and it will not continue before the user has clicked the dialogue box away. The status bar can be used for showing the module progress while executing.

More information about writing your own module can be obtained by studying the source files provided as appendix together with this document, and by reading the documentation supplied in the help file of Dynamic Imager. This help file contains a detailed and useful description of all the API functions.

5.2 System modules

This subchapter contains a detailed description of all the modules I have used and describes their most important functionality. ReadPGM and FaceNormalization were originally made by Lars Tellemann Sæther [6] and have only had minor changes to fit into our face recognition system. ShowDataset, WriteDataset, ReadDataset and FeedRead are built in Dynamic Imager modules which just need proper port and control parameter adjustments to function as wished. The Gabor module was developed just for visual verification of wavelet convolution. FaceGabor, GaborTrain and GaborRecognize have been developed from ground and are our systems three most important parts of the training of faces and recognition of faces.

5.2.1 ReadPGM

This module reads a PGM file from stable storage and converts it to Dynamic Imager dataset format. Path and filename of the PGM file is received through the “Input file location” port or by manually selecting the “File location” with a file browser. The “Output dataset” is a byte representation of the image found at the specified location. Input file location is divided into “Output filename” and “Output pathname” for use at other modules.

Selected input/output ports and control elements:

Input port	Input file location
Control element	File location
Output port	Output dataset [byte]
	Output filename
	Output pathname

5.2.2 ShowDataset

This module is a built-in module which shows to screen a Dynamic Imager dataset image. The “Input dataset” is delivered the module as a byte representation.

Selected input/output ports and control elements:

Input port	Input dataset [byte]

5.2.3 FaceNormalization

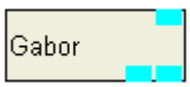
This module normalizes a face received through the “Input dataset”. First the face eye coordinates are found by looking for the “Input filename” in the eye coordinate file found at the “Coordinate file location”. The input image is then rotated so that the eye coordinates lie horizontal, and scaled so that the distance between the eyes is exactly 70 pixels. Then the image is then cut down to 150x130 pixels and an oval face mask found at “Mask file location” is applied to remove image background information. Finally we make a histogram equivalization and normalize the remaining pixel intensities with mean 0 and variance 1. The result is delivered both in byte and float format to “Output dataset”

Another usage of this module is possible if the boolean control element “Read all images in directory” is selected. Then the module will batch normalize all PGM images in an “Input directory” and write all the normalized NRM images to an “Output directory”. This functionality is not being used in our implementation.

Selected input/output ports and control elements:

Input port	Input filename
	Input dataset [byte]
Control element	Mask file location
	Coordinate file location
	Read all images in directory
	Input directory
	Output directory
Output port	Output dataset [float]
	Output dataset [byte]

5.2.4 Gabor

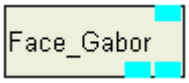


This module convolves the input image received through “Input dataset” pixel by pixel with a particular gabor wavelet with a specified rotation and scale. The control elements “Mask number” and “Odd wave” decide the shape of the wavelet used. The mask number can be selected between 0 and 15 predefined masks (4 different rotations and 4 different scales). The convolution result is delivered both as byte and float format to “Output dataset”. This module is only used to visually verify the correctness of the Gabor wavelet convolution.

Selected input/output ports and control elements:

Input port	Input dataset [float]
Control element	Mask number
	Odd wave
Output port	Output dataset [byte]
	Output dataset [float]

5.2.5 FaceGabor



This module convolves selected pixels in the input image received through “Input dataset” with all the gabor wavelets specified, also called a gabor jet. The selected pixels are chosen in an evenly spaced rectangular grid with 5 pixels apart in both x and y direction. The wavelets used have 4 different rotations and 4 different scales. The convolution result, a gabor jet image, is delivered both as byte and float format to “Output dataset”.

Selected input/output ports and control elements:

Input port	Input dataset [float]
Output port	Output dataset [byte]
	Output dataset [float]

5.2.6 GaborTrain

This module is a coordination module for the training of faces. The input is a gabor jet image that passes in through “Input dataset” and comes out through “Output dataset”. The corresponding filename received through “Input filename” is processed. A path is added and the extension is changed from PGM to MDS and the result sent to “Output file location”. The “Write dataset now” port is set to 1.

Selected input/output ports and control elements:

Input port	Input filename
	Input dataset [float]
Output port	Output file location
	Write dataset now
	Output dataset [byte]
	Output dataset [float]

5.2.7 WriteDataset

This module is a built-in module which writes a Dynamic Imager dataset image to file. If “Write dataset now” has value 1 then the image at “Input dataset” will be written to “Input file location”.

Selected input/output ports and control elements:

Input port	Write dataset now
	Input file location
	Input dataset [any type]

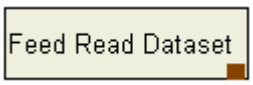
5.2.8 ReadDataset

This module is a built-in module which reads a Dynamic Imager dataset which has been stored to a file. “Input file location” decides which file should be read from stable storage. The dataset is delivered other modules through “Output dataset”.

Selected input/output ports and control elements:

Input port	Input file location
Output port	Output dataset [any type]

5.2.9 FeadReadDataset



This module is a built-in module. When the network is started the user can create a list of “Files” using the buttons “Add files” and “Delete files”. When the button “GO!” is pushed files will be sent subsequently to “Output file location”. The “Delay” factor, which has an initial value of 1000 milliseconds will decide at which rate the files are sent away.

Selected input/output ports and control elements:

Control element	Files [GO!, Add files, Delete files]
	Delay
Output port	Output file location

5.2.10 GaborRecognize

This module is a coordination module for the recognition of faces. It receives two datasets which has to be compared. The “Input test dataset” contains the test image that we want to recognize. The “Input database dataset” changes rapidly through the database of images stored on stable storage. Another required input parameter is “Input filename” which contains information about which database image is being compared to the test image. The output ports store the best matches found so far in the recognition process during network execution. “Output candidate1” contains the filename of the best database match, “Output candidate2” contains the filename of the second best database match and “Output candidate3” contains the filename of the third best database match. The module matches the incoming datasets by calculating a similarity measure of gabor jets. The similarity measure belonging to each of the candidates is stored in “Output value1”, “Output value2” and “Output value3” respectively. When a better match than the third best match is found the candidates are updated to reflect the overall three best matches.

Selected input/output ports and control elements:

Input port	Input filename
	Input database dataset [float]
	Input test dataset [float]
Output port	Output candidate1
	Output candidate2
	Output candidate3
	Output value1
	Output value2
	Output value3

5.3 System architecture

This subchapter contains a detailed description of how the modules were connected. First I will describe a simple normalization network and a simple Gabor wavelet convolution network. Then I will describe a training network used to create a database of gabor jet images, and the corresponding network for matching unknown images to the previously made database of trained images. Finally a simple network for comparing the similarity between two images is described.

5.3.1 Normalizing network

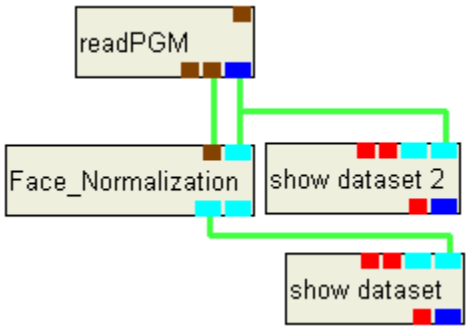


Figure 24: Image normalization network

The network above shows a simple network that normalizes a face. ReadPGM reads a selected image from stable storage and converts it to Dynamic Imager dataset format. The dataset is sent to the FaceNormalization module together with just the filename extracted from the full file location. FaceNormalization normalizes the incoming dataset as described and returns a normalized 130x150 pixel image. ShowDataset2 will show the original image, while ShowDataset will show the normalized image. The figure below shows a selected image and how it looks after normalization.

Figure 25: Image representation of (a) input dataset and (b) normalized dataset.

5.3.2 Gabor wavelet convolution network

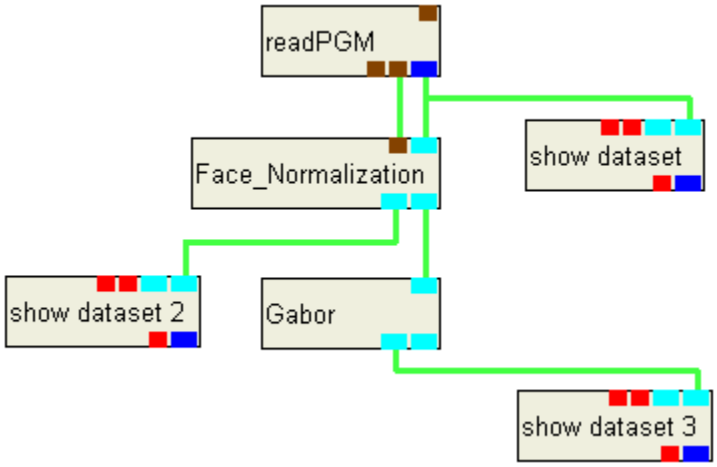


Figure 26: Gabor convolution network

This network extends the face normalization network described on the previous page. A float version of the normalized image is passed on to the Gabor module. The Gabor module makes a pixel by pixel convolution with a specified gabor wavelet mask. The user has a choice between applying a total of 32 predefined masks. When this is done the convoluted image is visually presented by ShowDataset3.

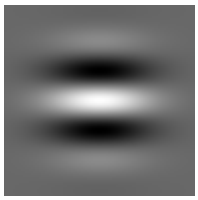
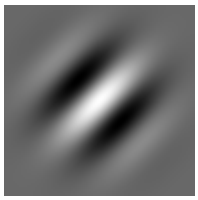
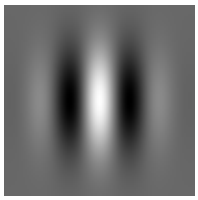
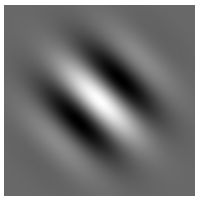
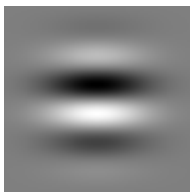
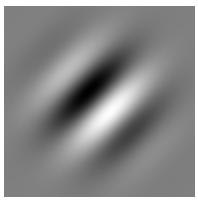
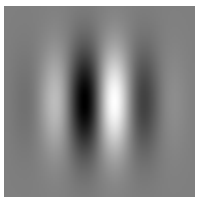
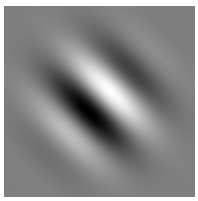
	0 degrees	**45 degrees**	**90 degrees**	**135 degrees**
Cosine Wave [Even Mask]				
Sine Wave [Odd Mask]				
95x95 pixels	Mask12	Mask13	Mask14	Mask15
47x47 pixels	Mask8	Mask9	Mask10	Mask11
23x23 pixels	Mask4	Mask5	Mask6	Mask7
11x11 pixels	Mask0	Mask1	Mask2	Mask3

Table 2: The 32 gabor wavelet masks and their mask numbers

The table above gives a visual view of what gabor wavelets looks like. The pictures show gabor wavelets of size 95x95 pixels with 4 different rotations (0, 45, 90 and 135 degrees). The first row of pictures shows the cosine gabor wavelets (even masks), and the second row of pictures shows the 90 degree phase shifted sine gabor wavelets (odd masks). Where the cosine wave has peaks the corresponding sine wave has

valleys, and where the cosine wave has valleys the corresponding sine wave has peaks. We use gabor wavelets not only with 4 different rotations, but also 4 different scales. The smaller gabor wavelet masks (47x47, 23x23 and 11x11 pixels) are not pictured here because they have the same properties and are only scaled down versions of the 95x95 pixel gabor wavelets. In the last four rows of the table above we have just named all the different masks mask0 to mask15 (4 rotations and 4 scales). Because there are both even and odd masks, that makes a total of 32 different convolution masks that can be applied to a pixel in the input image.

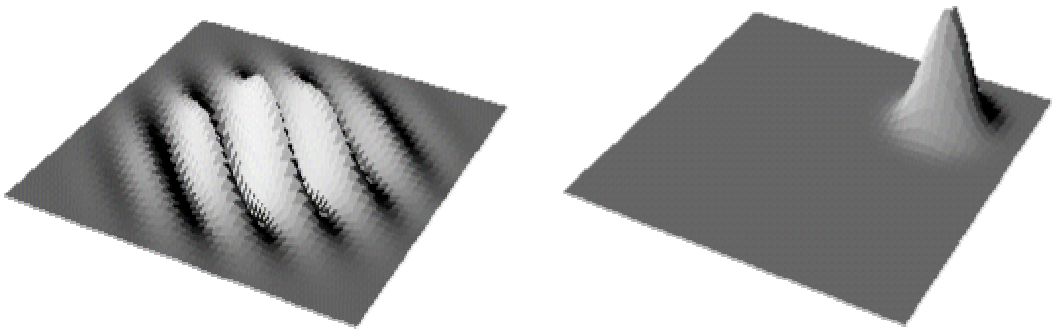


Figure 27: 3D representation of a gabor wavelet in (a) space domain (b) frequency domain

Figure 27 shows another way of looking at gabor wavelets. It shows what a gabor wavelet in space domain looks like when transformed to frequency domain.

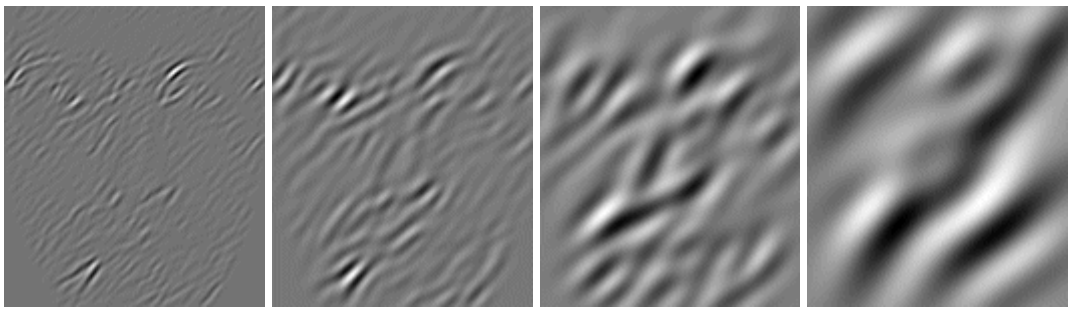


Figure 28: Convolution with 45 degree even masks with different mask sizes
(a) mask1 11x11 pixels (b) mask5 23x23 pixels (c) mask9 47x47 pixels (d) mask13 95x95 pixels

In the figure above we have used the same input image and normalized image as in the normalization example (Figure 25). We try to give a simple description of what is really going on. The example above shows convolutions with a 45 degree mask. 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.

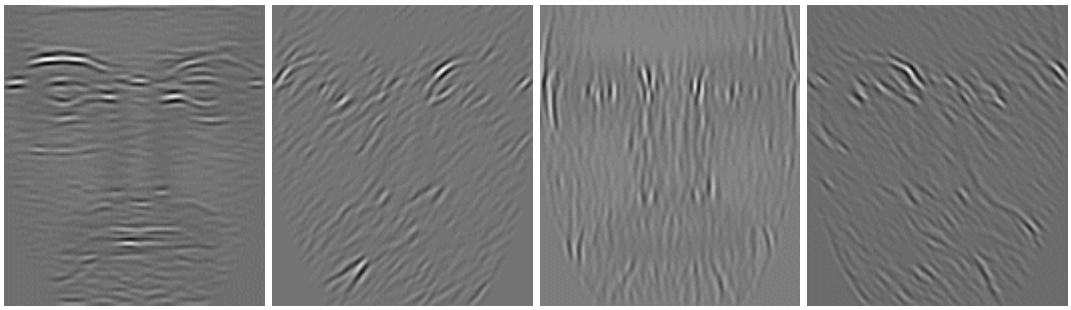


Figure 29: Convolution with even masks of size 11x11 pixels with different rotations (a) mask0 0 degrees (b) mask1 45 degrees (c) mask2 90 degrees (d) mask3 135 degrees

This second convolution example also uses the same input image and normalized image as in the normalization example (Figure 25). We show what happens if we try to change the rotation of the convolution mask. 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. One of the advantages of using gabor wavelets compared to other face recognition methods is that it to a great extent ignores the different light conditions in the input image and which makes it light invariant.

5.3.3 Face training network

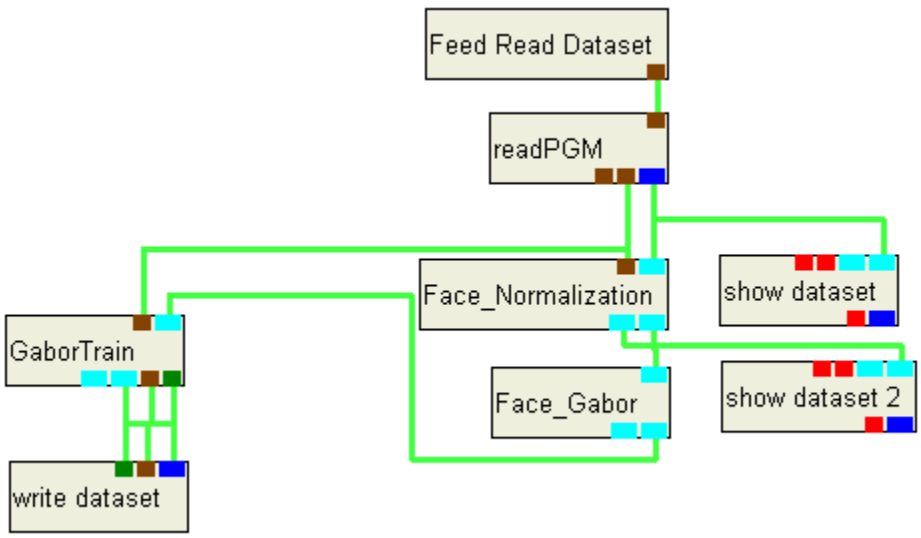


Figure 30: Face training network

We have divided our face recognition system in one face training network described here and one face recognition network described in the next subchapter. The training network trains a list of images selected with the FeedReadDataset module. The list of specified addresses is sent to the readPGM module which reads the images subsequently, converts them to Dynamic Imager dataset format and sends the datasets and their corresponding filename to the FaceNormalization module for normalization. Unlike the previous gabor wavelet convolution network the FaceGabor module does not a pixel by pixel convolution. A 5 pixel wide evenly spaced grid is placed onto the

normalized image dataset, and only the pixels on the grid are being sampled. In our case this means a grid consisting of 675 sampling pixels. For each of these 675 pixels we applied all 32 gabor wavelet masks specified earlier. This set of 32 gabor wavelet masks is called a gabor jet, and the set of 675 gabor jets representing the whole image is called the gabor jet image. A gabor jet gives a representation of how the pixel neighbourhood look like. By including all 32 gabor wavelets of different scales, different rotations and even or odd wave we try to catch all features which belong to a specified pixel. This makes this face recognition method both rotation and scale invariant. The result from the FaceGabor module is a gabor jet image consisting of 21600 pixel values (675 jets multiplied with 32 wavelets). This image is sent to the GaborTrain module which is the coordination module for training of faces. Every time it receives a gabor jet image from FaceGabor it uses WriteDataset to store it. The gabor jet image datasets are stored as 85 KByte MDS files on stable storage.

Table 3 gives a quick description of how we have stored data of the gabor jets in the MDS files. The 675 jets are placed sequential into the MDS file. Each of the jets contains an even wave coming first and an odd wave coming last. The even and odd waves are organized in a similar manner. First they are dived into 4 different scales and then each scale is divided into 4 different rotations. This small table gives a consistent layout of how we have stored a gabor image in a MDS file.

Jet																															
Even Wave (cosine part)																Odd Wave (sine part)															
1				2				3				4				1				2				3				4			
1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4

Table 3: Gabor jet data layout in the stored MDS file

One of the main purposes of the training network is to reduce the computational cost in the recognition stage. Because Dynamic Imager is an image processing simulating environment the computational cost is not in focus here. We could have implemented the recognition network such that it used a database of PGM images directly and skipped the training phase completely. Every comparison between the test image and the training image would then have to compute the gabor jet image during the recognition stage. This is not realistic in a real implementation, and is why we have not pursued this any further.

5.3.4 Face recognition network

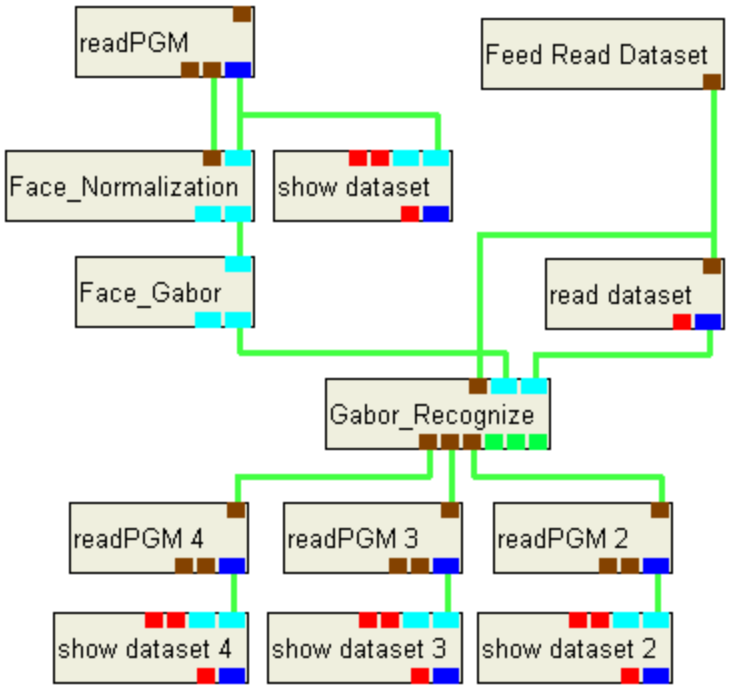


Figure 31: Face recognition network

The face recognition network uses the preprocessed MDS files stored by the training network. The FeedReadDataset module contains a list of all the MDS files trained and stored in the database. The stored gabor jet images are read at specified intervals one at a time by the ReadDataset module and sent to the GaborRecognize module. This is the coordination module for recognition of faces. The GaborRecognize module receives the trained gabor jet images subsequently and makes a gabor jet comparison with the test image for each trained image. The test image is the unknown image we want to identify, and is not part of the MDS database. Its gabor jets have to be calculated in the exact same manner as the test images for comparison.

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 29). We later will show some of our results when using this simple but still powerful similarity measure.

Equation 29: $$S_a(J,J') = \frac{\sum_j a_j a'_j}{\sqrt{\sum_j a_j^2 a'^2_j}}$$

The magnitude/amplitude a_j and the phase ϕ_j is calculated on the basis of the even (cosine) and odd (sine) wavelet pair just as with complex numbers (Equation 30 and Equation 31). Our similarity measure needs the amplitude of 16 different combinations (4 rotations and 4 scales) for every jet.

Equation 30: $$a_j = \sqrt{\text{EvenWave}_j^{\,2} + \text{OddWave}_j^{\,2}}$$

Equation 31: $$\phi_j = \arctan\left(\frac{\text{OddWave}_j}{\text{EvenWave}_j}\right)$$

To get a global similarity measure between two face images we have just summed up all the similarity measures and divided on the total number of jets which is 675. This gives an overall measure of how close two faces are to each other. The GaborRecognize module stores the three values closest up to 100% and their filenames on the output ports. A similarity value of 100% indicates that the training image and test image are identical. If a better match is found during network execution the three values and candidate filenames are updated. When all gabor training images have been compared with the test gabor image the values and filenames of the three best trained matches will remain at the output ports. ReadPGM 2, 3 and 4 retrieve the original candidate images by converting them from PGM to Dynamic Imager dataset format and letting ShowDataset 2, 3 and 4 present the results visually to the user.

5.3.5 Face comparison network

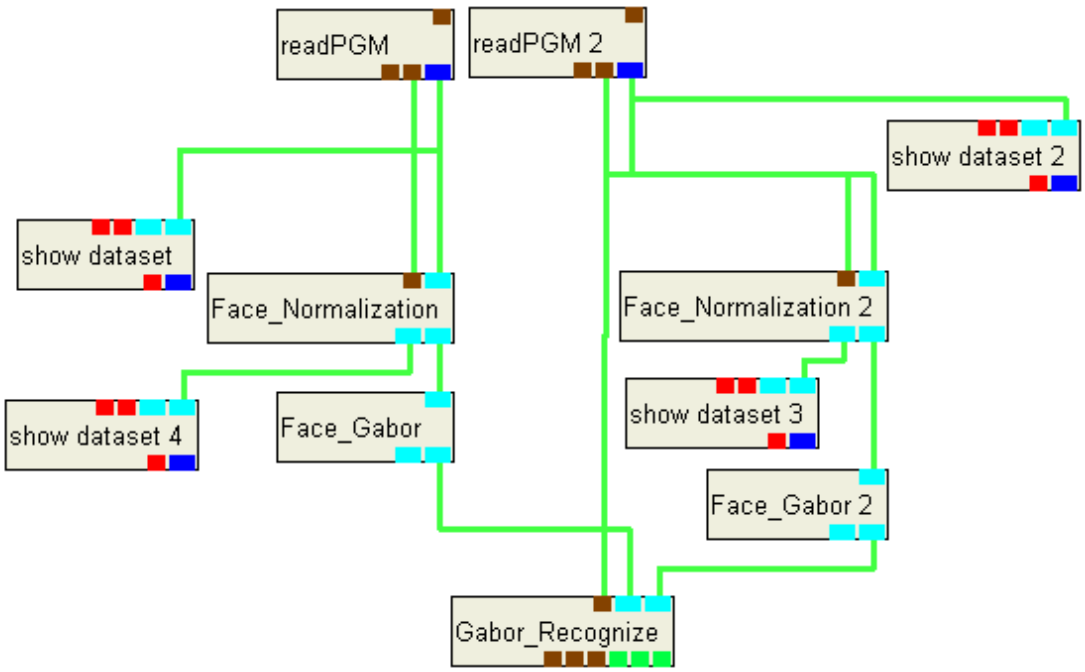


Figure 32: Face comparison network

This face comparison network has just been used for finding the similarity measure between two pictures. The test image process is just duplicated on both input ports of the GaborRecognize module.

6 Results

This chapter contains information about the FERET image database which we have been using extensively. We also have done some tests to find out how good our implementation is and especially find out why recognition fails in some cases.

6.1 The FERET Database

The FERET database contains a total of 14126 images from 1199 different individuals, and is one of the largest face databases collected. The FERET program ran from 1993 to 1997. Its primary mission is to help testing and evaluating different face recognition algorithms, and be able to compare them by using the same dataset. The FERET database is open to all non-commercial interests and can be obtained on two CD-ROMs. The FERET images are all eight-bit greyscale images of human heads with views ranging from frontal to left and right profiles. The naming convention for the FERET images in the distribution is of the form nnnnnxxfffq_yymmdd.ext where:

1. nnnnn is a five digit integer that uniquely identifies the subject individual.

2. xx is are two lowercase characters that indicate the kind of image.

Letter Code	Pose/ Angle	Description	Number of images	Number of subjects
fa	0	Regular face expression	1762	1010
fb	0	Alternative face expression to fb	1518	1009
ba	0	Regular face expression	200	200
bj	0	Alternative expression to ba	200	200
bk	0	Different illumination to ba	200	200
bb	+60	Subject faces to his left which is the photographer's right	200	200
bc	+40		200	200
bd	+25		200	200
be	+15		200	200
bf	-15	Subject faces to his right which is the photographer's left	200	200
bg	-25		200	200
bh	-40		200	200
bi	-60		200	200
ql	-22.5	Quarter left and right	763	508
qr	+22.5		763	508
hl	-67.5	Half left and right	1246	904
hr	+67.5		1298	939
pl	-90	Profile left and right	1318	974
pr	+90		1342	980
ra	+45	Random images.	322	264
rb	+10		322	264
rc	-10		613	429
rd	-45		292	238
re	-80		292	238

Table 4: Table of the different FERET images

3. fff is a set of three binaries. For instance if the flag has a value 1 it has been histogram adjusted.

4. q is not always present. The meanings are as follows:

   a. Glasses worn
   b. Different hair length
   c. Glasses worn and different hair length
   d. Computer scaled and histogram adjusted
   e. Clothing has been computer retouched
   f. Reduced brightness with 40%
   g. Reduced brightness with 80%
   h. Reduced image size with 10%
   i. Reduced image size with 20%
   j. Reduced image size with 30%

5. yymmdd is the date the picture was taken in year, month and day format.

6. The filename extension is TIF. The images on the CD-ROM carry an additional BZ2 suffix that indicates that the files have been losslessly compressed using the free bzip2 compressor.

We decided to use only a subset of the images supplied in the FERET database. The FERET database comes with an eye coordinate file COORDS3816.TXT which does not include eye coordinates for all of the pictures on the CD-ROMs. We could have punched in a lot of eye coordinates manually, but chose to select pictures which already had predefined eye coordinates in the supplied coordinate file. A total of 99 subject individuals and 303 images were selected for use as training set in our performance tests.

All images in the FERET database were unzipped and converted from TIFF to PGM format. The TIFF images are losslessly compressed to preserve all the detailed image information. A lossy compressed image format like JPEG looks fine at high resolution, but at low resolution you can see that a lot of details have been lost. Unzipping the BZ2 zipped files was done on a Redhat Linux based machine which had bunzip installed. Conversion of all the images from TIFF format to PGM format was done using a Windows based batch conversion utility called IrfanView. One of the big advantages with the PGM (Portable Gray Map) format is that it is many times faster to read and write. Time will not be tested because all the implementation is done in a simulation environment. The FeedReadDataset module and its delay time limits the recognition time. A delay time of 500 ms times 300 images totals 3,5 minutes running time for comparing an image to 300 other trained images. In a real system one has to consider running time more seriously and the PGM format would be valuable for reducing this.

6.2 Face recognition tests

This subchapter contains information about the different tests that have been run, with special focus on what went wrong and how to make things go wrong. The ideology is that nothing is learnt if everything goes well.

6.2.1 Initial test run

Figure 33: Correct identification. Test image (left) Best match (middle) Second best match (right)

First we wanted to see how good our implementation was. This was done by comparing all 303 images to the other images. Recognition of an image was done by removing that image from the training set in the FeedReadDataset module, comparing it to all the remaining images and see what is the best match. Figure 33 shows how the first subject individual in the dataset (00001) is correctly matched with the trained image having a different facial expression. The similarity measure was a 97,5068% match, while the second best match (00012) had a similarity measure of 96,3096%. It can maybe look like the similarity measures are very close since they all report over 90% match, but the following tests show good results in comparing images from the FERET database even with this simple measure. Faces are in general very similar, and if we had tried to test something quite different, like an image of a horse, we would certainly get a similarity measure beneath 90%.

Figure 34: False identification. Test image (left) Best match (middle) Second best match (right)

Sometimes the identification done is wrong. In Figure 34 is a good example of false recognition. When trying to recognize a test image subject individual (00003) both the best match (00070) and the second best match (00080) is wrong. The similarity measure for best match was 97,0535% and second best match 96,4306%. The highest similarity measure for one of the images that should have been recognized was only 94,9714%. It is not always easy to say what went wrong. One of the reasons can be that the database training images were too different compared to the test image. This is not the reason in this case. The alternative face expression looks not that different. I suspect that it can have something to do with the way we compare gabor jets. The problem comes when the jets belonging to the same area are too far away from each other during comparison. This could maybe have been avoided by using the phase information as well as the magnitude information when comparing the gabor jets. Also a better normalizing module could have helped making the images more consistent.

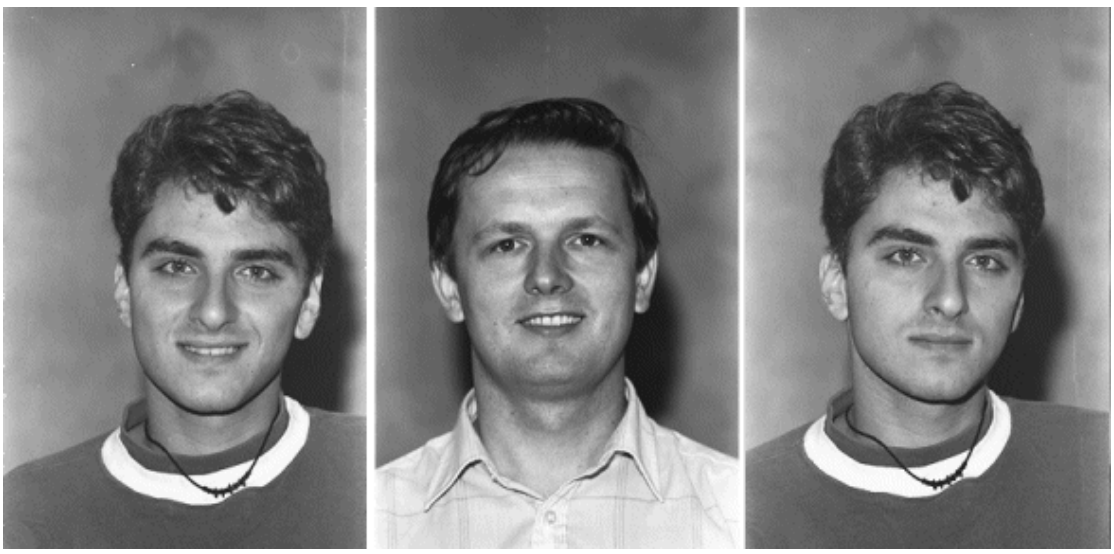


Figure 35: False identification. Test image (left) Best match (middle) Correct match (right)

Figure 35 also shows a false identification of a test image (00017). The similarity rates were very close with a best match (00004) of 96,2757% and a correct identification measure of 96,0146%. Here we can somehow see what went wrong. The training image (correct match) has a slightly different angle compared to the test image, which makes the recognizer pick a wrong subject facing forward and smiling as the test image does. If the recognizer should be able to recognize people from different angles, it is clearly not enough with only one image in the training set. For a full 180 degree frontal recognition ranging from left profile to frontal to right profile it is recommended to have at least 7 images. Better recognition rates can be achieved with maybe as many as 10 or 20 images.

For all 303 images of 99 subject individuals we got 18 false identified and 285 correct identified. This means a recognition rate of 94,1%. Bear in mind that this was done on the FERET database and not in a natural environment. Most of the recognized images are taken on the same day, with the same illumination and camera, and have only minor differences in face expression.

6.2.2 Increasing the difficulty level

Figure 36: Recognizing people with sun glasses

In this experiment we removed the database images of a particular individual (00011/00012) leaving only one picture where he wears sun glasses. There seemed to be no severe difficulties recognizing the database picture. The similarity measure reached 96,9460%, a little bit lower score than 99,4259% in the case where the training image also has glasses, but it was enough to become the best match. It does not look like sun glasses interfere that much when looking at the overall match.

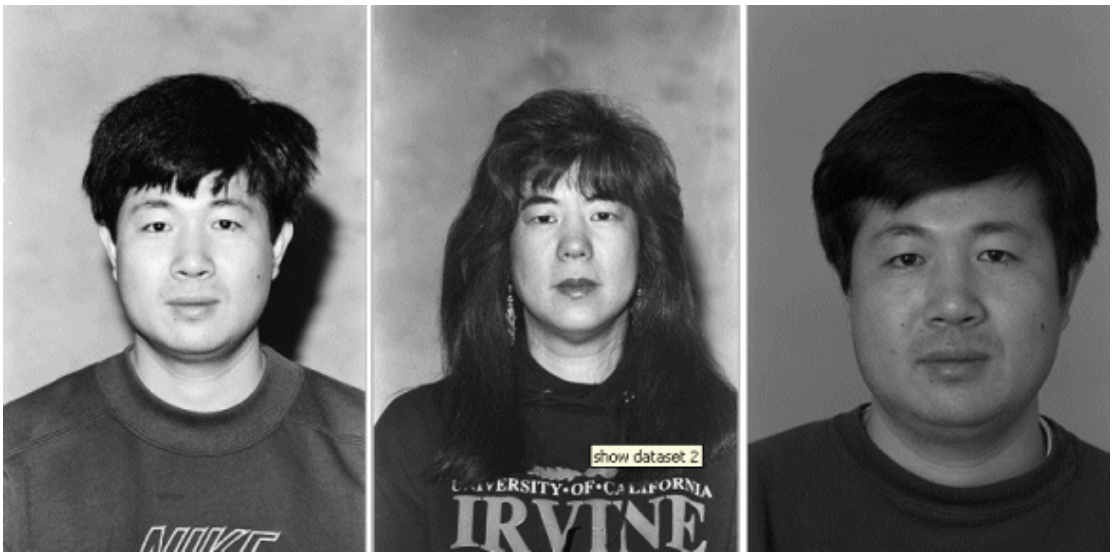


Figure 37: Test image (left) False identified (middle) Correct identified (right)

Another interesting experiment is looking at the aging problem. People change appearance over time, and this fact decreases the recognition performance drastically. The figure above shows a test image (00029) taken in 1993 which in comparison with all images is correctly identified with its corresponding image having a different facial expression also taken in 1993. The similarity measure showed a 99,8568% match. To make it a little bit more difficult we removed all images of the subject individual leaving only one picture from 1996. This resulted in an expected false recognition (middle picture) with similarity measure of 96,6829%. We can observe that the woman identified has Asian facial features like the guy we wanted to identify, but it is still clearly a false identification. If we leave 21 pictures in the training database, all from 1994 or 1996, a picture from 1996 (right picture) is correctly identified as being the test subject. The similarity measure reaches 97,3302%. If you want to recognize people over time you have to have many pictures in the database to overcome the aging problem. It looks like a minimum of 20 pictures must be present in the training database.

6.3 Future work

Because of the limited time of this project, not everything could be investigated as thoroughly as we wanted to. In this subchapter we will describe some of the most interesting extensions to this project.

Some of the following topics can be investigated further:

1. Improve the FaceNormalization module. We have encountered some minor problems especially with pictures where the head is rotated slightly horizontally. One could imagine that the FaceNormalization module also considered such head rotation, and not only the vertical rotation (alignment of the eyes).

2. Improve the FaceGabor module. Gabor wavelets come in pairs and have different sizes and rotations. We selected to have 4 rotations and 4 sizes. It could have been interesting to further investigate the use of different sizes, number of sizes and number of rotations used. Is there an optimal choice of wavelet pairs?

3. Improve the fiducial face grid. We used a regular spaced rectangular grid for sampling the gabor jets. What effect has the number of fiducial points and how does spacing affect recognition. One could also imagine sampling a from a face graph where the fiducial points are known facial landmarks like eyes, nose, earlobes etc. In order to find fiducial points in new faces one needs a general representation model called a face bunch graph (FBG). This is described earlier and consists of nodes with jets representing different appearances of facial features. The face matching procedure normally uses heuristic methods because graph matching is an NP-complete problem.

4. Improve the similarity measure. Maybe there is a more clever way of comparing jets. We decided to ignore phase information. It could be interesting to see the effect of having phase information. Also our comparison is made without any consideration to any of the neighbour jets. A little search for the closest match between neighbour jets could be a good idea to improve recognition rates and make it more invariant to small horizontal rotation changes.

5. Look at using other image classifiers in parallel. It could be interesting looking at different segmentation issues, using stereo vison or colour information and other even more simple techniques to improve the face recognition performance.

6. Look at using gabor jets as input to a feed forward neural network. Neural networks deliver fast recognition, but require much training especially on large training databases.

7. Make help files. Due to time limits there has not been time for making any help files for the developed modules.

7 Conclusion

One of the big advantages with face recognition is that it is a very natural procedure, and that it is a non-intrusive biometric method that can be done at some distance. Although it is very easy for humans to spot, recognise, and identify human faces, there is still no automatic system which solves this task reliably. Several difficulties due to variable orientations, sizes and changing light conditions still have to be overcome. Facial expressions, beards, scarf, glasses, make-up, disguises and aging make the process even harder. Reliable face and facial feature detection and tracking in complex scenes still remains a challenge.

The results of a modern face recognition system with a database of above 1000 individuals are affected by light conditions and by time. Pictures taken on the same day with under the same light conditions and same camera settings can typically achieve a recognition rate of 95%. If the light conditions and camera settings change the recognition rate drops to maybe 80%. Pictures taken a year later will maybe have only a 50% recognition rate. And all these rates come from face recognition in controlled environments. In uncontrolled real life environments, the rates are probably even lower.

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 normalization process is relatively unaffected by different rotations, scales and light conditions. To recognize people by their profile, the database of trained images needs to have profile images stored.

We found that having many trained images in the database of the same subject individual increases the chances of recognizing the test image. Another important issue is the choice of similarity measure. Improvements can be made to our implementation to increase the rate of correct versus false identifications.

The most interesting future work topics are related to face bunch graphs and improving the similarity measure. It could also be interesting looking at using other classifiers in parallel or using gabor jets as input to neural networks.

8 References

8.1 Litterature

[1] "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection", Peter N. Belhumeur, Joao P. Hespana and David J. Kriegman (1997)

[2] "Using Discriminant Eigenfeatures for Image Retrieval", D. L. Swets and J. Weng (1996)

[3] "The Use of Multiple Measurements in Taxonomic Problems", R. A. Fisher (1936)

[4] "Discriminant Analysis of Principal Components for Face Recognition", W. Zhao, A. Krishnaswamy, R. Chellappa, D. L. Swets and J. Weng (1998)

[5] "Discriminant Analysis for Recognition of Human Face Images", K. Etemad and R. Chellappa (1997)

[6] "Ansiktsgjenkjenning", Lars Tellemann Sæther (2002)

[7] "Analysis of PCA-Based and Fisher Discriminant-Based Image Recognition Algorithms", Wendy S. Yambor, Technical Report Colorado State University (2000)

[8] "Feature Extraction from faces using deformable templates", A. L. Yuille, D. S. Cohen, and P. W. Hallinan (1989)

[9] "Face Recognition Using Eigenfaces", Matthew A. Turk and Alex P. Pentland (1991)

[10] "Picture processing by computer complex and recognition of human faces", T. Kanade (1973)

[11] "Face Recognition: Features versus Templates", R. Brunelli and T. Poggio (1993)

[12] "Eigen Light-Fields and Face Recognition Across Pose, R. Gross et al. (2002)

[13] "The plenoptic function and elements of early vision", E. Adelson and J. Bergen (1991)

[14] "Light field rendering", M. Levoy and M. Hanrahan (1996)

[15] "PCA vs. ICA: A comparison on the FERET data set", Kyungim Baek, Bruce A. Draper, J. Ross Beveridge and Kai She (2000)

[16] “Image Processing, Analysis and Machine Vision”, Milan Sonka, Vaclav Hlavac, Roger Boyle (1999)

[17] “Face Recognition by Elastic Bunch Graph Matching”, Laurenz Wiscott, Jean-Marc Fellous, Norbert Krüger and Christoph von der Malsburg (1996)

[18] “Face Recognition Using Neural Networks”, Nazish Jamil, Samreen Iqbal and Naveela Iqbal (2001)

[19] “Automatic Face Recognition System Using Neural Networks”, El-Bakry and Abo-Elsoud, Kamel (2000)

[20] “Understanding Neural Network as Statistic Tools, Brad Warner and Manavendra Misra (1996)

[21] “Face Detection and Recognition Using Hidden Markov Models”, Ara V. Nefian and Monson H. Hayes (1998)

[22] “A Hidden Markov Model-Based Approach for Face Detection and Recognition”, Ara V. Nefian (1998)

[23] “Hidden Markov for Face Recognition”, Nefian and Hayes (1998)

[24] ”Low-Dimensional Procedure for the Characterization of Human Faces”, L. Sirovich and M. Kirby (1987)

[25] “Subspace Methods for Robot Vision”, Shree K. Nayar, Sameer A. Nene and Hiroshi Murase (1995)

[26] “View-Based and Modular Eigenspaces for Face Recognition”, Alex P. Pentland, Baback Moghaddam, Thad Starner (1994)

[27] “Gjenkjenning av Ansikter”, Line Eikvil (2001)

[28] ”Biometric Systems: A Face Recognition Approch”, Erik Hjelmås (2000)

[29] “Visual Properties of Neurons in inferotemporal cortex of the macaque”, C. Gross, C. Rocha-Miranda and D. Bender (1972)

[30] “Face Recognition: A Summary of 1995-1997”, Thomas Fromherz (1997)

[31] “Face Recognition: A Critical Look at Biologically-Inspired Approaches”, Gita Sukthankar (1999)

[32] “Hybrid Hidden Markov Model for Face Recognition”, Hisham Othman and Tyseer Aboulnasr (2000)

[33] “Spoken Language Processing: A Guide to Theory, Algorithm and System Development”, Xuedong Huang, Alex Acero, Hsiao-Wuen Hon and Raj Reddy (2001)

[34] "Low complexity 2-D Hidden Markov Model for Face Recognition", Hisham Othman and Tyseer Aboulnasr (2000)

[35] "A Gabor Feature Classifier for Face Recognition", Chengjun Liu and Harry Wechsler (2001)

[36] "Gabor Feature Based Classification Using the Enhanced Fisher Linear Discriminant Model for Face Recognition", Chengjun Liu (2002)

[37] "A simplified second-order HMM with application to face recognition", Hisham Othman and Tyseer Aboulnasr, (2001)

[38] "Memory-Based Face Recognition for Visitor Identification", Terence Sim, Rahul Sukthankar, Matthew Mullin and Shumeet Baluja (1999)

[39] "Beyond Euclidean Eigenspaces: Bayesian Matching for Visual Recognition", Baback Moghaddam and Alex P. Pentland (1998)

[40] "Face Recognition: From Theories to Applications", L. C. Jain, U. Halici, I. Hayashi, S. B. Lee and S. Tsutsui (1998)

[41] "Toward Automatic Simulation of Aging Effects on Face Images", Andreas Lanitis, Chris J. Taylor and Timothy F. Cootes (2002)

[42] "Detecting Faces in Images: A Survey", Ming-Hsuan Yang, David J. Kriegman and Narendra Ahuja (2002)

[43] "Principal Manifolds and Probabilistic Subspaces for Visual Recognition", Baback Moghaddam (2002)

[44] "Face Detection in Color Images", Rein-Lien Hsu, Mohamed Abdel-Mottaleb and Anil K. Jain (2002)

8.2 Hyperlinks

We decided not to refer to any hyperlinks inside this document, because most of them change and disappear over time. Instead we list some of the most important links used here. The reader is encouraged to continue reading on these web pages for additional material about face recognition and other topics.

8.2.1 General

- Face recognition homepage : http://www.cs.rug.nl/~peterkr/FACE/face.html
- Face detection homepage : http://home.t-online.de/home/Robert.Frischholz/face.htm
- Face recognition demo page : http://www-white.media.mit.edu/vismod/demos/facerec/
- Evaluation of Face Recognition Algorithms : http://www.cs.colostate.edu/evalfacerec/index.html
- Computational Approaches to Face Recognition : http://www.ski.org/CWTyler_lab/CWTyler/PrePublications/ARVO/1998/FaceRecog/
- Face Recognition Bibliography : http://www.cnl.salk.edu/~wiskott/Bibliographies/FaceRecognition.html
- Marquard Beauty Analysis : http://www.beautyanalysis.com/
- Leiden: Face Detection and Recognition : http://www.wi.leidenuniv.nl/home/lim/face.detection.html
- Advanced Topics in Computer and Human Vision : http://www.wisdom.weizmann.ac.il/~ronen/course/spring01/Course.html
- Grundsätliche Untersuchung von Bildverarbeitungsalgorithmen zur Gesichtserkennung : http://www.markus-hofmann.de/
- Instiute for Neuroinformatik: Computer Vision : http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/computerVision/contents.html
- Multiplied functions unify shapes of ganglion-cell receptive fields in retina of turtle : http://www.journalofvision.org/2/3/1/article.html
- The FERET Database : http://www.itl.nist.gov/iad/humanid/feret/

8.2.2 Commercial Interests

- FaceKey Corporation : http://www.facekey.com/
- Identix FaceIT : http://www.faceit.com/
- Biometrics Consortium : http://www.biometrics.org/
- BioAPI Consortium : http://www.bioapi.org/

8.2.3 Hidden Markov Models

- Hidden Markov Models : http://home.ecn.ab.ca/~jsavard/crypto/co041003.htm
- Explanation of the 2D DCT (Discrete Cosine Transform) : http://www-mtl.mit.edu/Courses/6.095/spring-00/unit3/dct.html

8.2.4 Neural Networks

- Multilayer Perceptron in Face Recognition : http://www.electronicsletters.com/papers/2001/0020/paper.asp

8.2.5 Eigenfaces and Fischerfaces

- Face Recognition and Detection Using Eigenfaces : http://www.cc.gatech.edu/classes/cs7322_97_spring/participants/Gonzalez/final/node1.html
- Eigenfaces Group : http://www.owlnet.rice.edu/~elec301/Projects99/faces/index.html
- Modelling in Applied Mathematics : http://amath.colorado.edu/courses/4380/All/contents.html
- Eigenvalue : http://mathworld.wolfram.com/Eigenvalue.html
- Wendy S Yambor : http://www.cs.colostate.edu/~vision/html/projects/yambor/thesis.htm

8.2.6 Wavelets

- Wavelets : http://www.cosy.sbg.ac.at/~uhl/wav.html
- Wavelets : http://www.cs.rug.nl/~wink/Wavelets/Wavelets_En.html
- Wavelet Resources : http://www.mathsoft.com/wavelets.html
- An Introduction to Wavelets : http://www.amara.com/IEEEwave/IEEEwavelet.html

8.2.7 Graph Matching

- Face Recognition by Elastic Bunch Graph Matching : http://www.cnl.salk.edu/~wiskott/Projects/EGMFaceRecognition.html
- Graph Matching and Comparison : http://www-users.cs.york.ac.uk/~sara/reference/graphmatch/

8.2.8 Implementation

- C++: What and Why? : http://hepwww.ph.qmul.ac.uk/mcps/intro.shtml
- PGM files : http://www.math.iastate.edu/burkardt/data/pgm/pgm.html
- IrfanView : http://www.irfanview.com/

9 Indexes

9.1 Figures

9.2 Tables

10 Appendix

10.1 FaceGabor.cpp

```
#include <mtip.h>
#include <string.h>
#include "GLOBAL.H"
#include "DATASETCOMMON.H"

#define MY_HELP_FILE   0 // TODO Change to path&name to your help file myhelpfile.hlp
#define MY_HELP_TOPIC  0 // TODO Change to help topic no of the help for this module

// External Gabor Filter Masks
extern float apply_mask0even(float*,int,int,int,int);
extern float apply_mask1even(float*,int,int,int,int);
extern float apply_mask2even(float*,int,int,int,int);
extern float apply_mask3even(float*,int,int,int,int);
extern float apply_mask4even(float*,int,int,int,int);
extern float apply_mask5even(float*,int,int,int,int);
extern float apply_mask6even(float*,int,int,int,int);
extern float apply_mask7even(float*,int,int,int,int);
extern float apply_mask8even(float*,int,int,int,int);
extern float apply_mask9even(float*,int,int,int,int);
extern float apply_mask10even(float*,int,int,int,int);
extern float apply_mask11even(float*,int,int,int,int);
extern float apply_mask12even(float*,int,int,int,int);
extern float apply_mask13even(float*,int,int,int,int);
extern float apply_mask14even(float*,int,int,int,int);
extern float apply_mask15even(float*,int,int,int,int);

extern float apply_mask0odd(float*,int,int,int,int);
extern float apply_mask1odd(float*,int,int,int,int);
extern float apply_mask2odd(float*,int,int,int,int);
extern float apply_mask3odd(float*,int,int,int,int);
extern float apply_mask4odd(float*,int,int,int,int);
extern float apply_mask5odd(float*,int,int,int,int);
extern float apply_mask6odd(float*,int,int,int,int);
extern float apply_mask7odd(float*,int,int,int,int);
extern float apply_mask8odd(float*,int,int,int,int);
extern float apply_mask9odd(float*,int,int,int,int);
extern float apply_mask10odd(float*,int,int,int,int);
extern float apply_mask11odd(float*,int,int,int,int);
extern float apply_mask12odd(float*,int,int,int,int);
extern float apply_mask13odd(float*,int,int,int,int);
extern float apply_mask14odd(float*,int,int,int,int);
extern float apply_mask15odd(float*,int,int,int,int);

int FaceGabor_MDF(HMTIPMOD hMod)
{

  INPUT_DATASET("Dataset Input (Normalized raster)", 2, 1, DST_FLOAT, CONN_REQUIRED);
  CONTROL_BOOL("Verbose info",0);
  OUTPUT_DATASET("Dataset Output (Convolved)", 2, 1, DST_FLOAT, CONN_OPTIONAL);
  OUTPUT_DATASET("Dataset Output (Convolved)", 2, 1, DST_BYTE, CONN_OPTIONAL);
  return MRV_SUCCESS;
}

int FaceGabor_MIF(HMTIPMOD hMod,
                  Dataset_float *in_ds,
                  bool ctrl_verbose,
                  Dataset_float **out_float,
                  Dataset_byte **out_byte)
{
  float *image, *output;

  if(!in_ds) {
    mmsg_Report("Error","No data on input.");
    return MRV_FAILURE;
  }
  else image = DATA_PTR(in_ds);
```

```
// Creating Gabor Jet Image
  int x,y;
  int n = 0;
  output = (float*)malloc(sizeof(float)*50000); // (25 * 27) samples * 32 verdier

  for(x = 4; x < NRM_WIDTH-4; x+=5) {
    for(y = 9; y < NRM_HEIGHT-4; y+=5) {

      output[n++] = apply_mask0even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask1even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask2even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask3even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask4even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask5even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask6even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask7even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask8even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask9even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask10even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask11even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask12even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask13even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask14even(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask15even(image, NRM_WIDTH,NRM_HEIGHT, x, y);

      output[n++] = apply_mask0odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask1odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask2odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask3odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask4odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask5odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask6odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask7odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask8odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask9odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask10odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask11odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask12odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask13odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask14odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
      output[n++] = apply_mask15odd(image, NRM_WIDTH,NRM_HEIGHT, x, y);
    }
  }

  writeDatasetFloat(out_float, output, 160, 135, ctrl_verbose);
  writeDatasetByte(out_byte, output, 160, 135, ctrl_verbose);
  return MRV_SUCCESS;
}

REGISTER_MTIP_MODULE("Face_Gabor|Atle",FaceGabor_MDF,FaceGabor_MIF,0,MY_HELP_FILE,MY_H
ELP_TOPIC);
```

10.2 GaborTrain.cpp

```
#include <mtip.h>
#include <string.h>
#include "DATASETCOMMON.H"
#include "GLOBAL.H"

#define MY_HELP_FILE   0 // TODO Change to path&name to your help file myhelpfile.hlp
#define MY_HELP_TOPIC  0 // TODO Change to help topic no of the help for this module

int GaborTrain_MDF(HMTIPMOD hMod)
{
  INPUT_DATASET("Dataset Input (Normalized raster)", 2, 1, DST_FLOAT, CONN_OPTIONAL);
  INPUT_TEXT("Input Filename",0);
  CONTROL_BOOL("Verbose info",0);
  OUTPUT_INT("Write",0);
  OUTPUT_TEXT("Output Filename",0);
  OUTPUT_DATASET("Dataset Output (Gabor raster)", 2, 1, DST_FLOAT, CONN_OPTIONAL);
  OUTPUT_DATASET("Dataset Output (Gabor)", 2, 1, DST_BYTE, CONN_OPTIONAL);

  return MRV_SUCCESS;
}
```

```
int GaborTrain_MIF(HMTIPMOD hMod,
                   Dataset_float *in_ds,
                   char * in_filename,
                   bool ctrl_verbose,
                   int * out_write,
                   char **out_filename,
                   Dataset_float **out_float,
                   Dataset_byte **out_byte)
{
  float *output;

  if(!in_ds) {
    mmsg_Report("Error","No data on input.");
  return MRV_FAILURE;
  } else {
    output = DATA_PTR(in_ds);
    writeDatasetFloat(out_float, output, 160, 135, ctrl_verbose);
    writeDatasetByte(out_byte, output, 160, 135, ctrl_verbose);
  }

  char* endname = strrchr(in_filename,'.');
  if(endname) endname[0] = '\0'; //terminate string

  *out_filename = (char*) malloc(sizeof(char)*MAX_FILENAME_LENGTH);

  char file[MAX_FILENAME_LENGTH];
  strcpy(file,DEFAULT_MDS_DIR);
  strcat(file,in_filename);
  strcat(file,".mds");
  strcpy(*out_filename,file);

  *out_write = 1;

  return MRV_SUCCESS;
}

REGISTER_MTIP_MODULE("GaborTrain|Atle",GaborTrain_MDF,GaborTrain_MIF,0,MY_HELP_FILE,MY
_HELP_TOPIC);
```

10.3 GaborRecognize.cpp

```
#include <mtip.h>
#include <math.h>
#include <string.h>
#include "GLOBAL.H"

#define MY_HELP_FILE   0 // TODO Change to path&name to your help file myhelpfile.hlp
#define MY_HELP_TOPIC  0 // TODO Change to help topic no of the help for this module

#define JETS 675
#define SCALES 4
#define ROTATIONS 4

#define JETWIDTH 32
#define SCALEWIDTH 4

int GaborRecognize_MDF(HMTIPMOD hMod)
{
  INPUT_DATASET("Dataset Input (Gabor Test Image)", 2, 1, DST_FLOAT, CONN_OPTIONAL);
  INPUT_DATASET("Dataset Input (Gabor Data Image)", 2, 1, DST_FLOAT, CONN_OPTIONAL);
  INPUT_TEXT("Filename",CONN_OPTIONAL);

  CONTROL_BOOL("Verbose info",0);

  OUTPUT_FLOAT("Similarity 1",CONN_KEEPDATA);
  OUTPUT_FLOAT("Similarity 2",CONN_KEEPDATA);
  OUTPUT_FLOAT("Similarity 3",CONN_KEEPDATA);
  OUTPUT_TEXT("Filename 1",CONN_KEEPDATA);
  OUTPUT_TEXT("Filename 2",CONN_KEEPDATA);
  OUTPUT_TEXT("Filename 3",CONN_KEEPDATA);

  return MRV_SUCCESS;
}
```

```
int GaborRecognize_MIF(HMTIPMOD hMod,
                       Dataset_float *in_dstest,
                       Dataset_float *in_dsdata,
                       char * in_file,
                       bool ctrl_verbose,
                       int ctrl_debug,
                       float *out_similarity1,
                       float *out_similarity2,
                       float *out_similarity3,
                       char **out_file1,
                       char **out_file2,
                       char **out_file3)
{
  float *in_test, *in_data;

  if(!in_dstest) {
    mmsg_Report("Error","No data on input.");
    return MRV_FAILURE;
  }
  else in_test = DATA_PTR(in_dstest);

  if(!in_dsdata) {
    mmsg_Report("Error","No data on input.");
    return MRV_FAILURE;
  }
  else in_data = DATA_PTR(in_dsdata);

  int jet, scale, rot;

  float re_test, im_test, re_data, im_data, mag_test, mag_data;
  float sum1, sum2a, sum2b;
  float similarity = 0;

  for (jet = 0; jet < JETS; jet++) {              // 675 Jets

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

    for (scale = 0; scale < SCALES; scale++) { // 4 Scales
      for (rot = 0; rot < ROTATIONS; rot++) { // 4 Rotations

        re_test = in_test[rot+scale*SCALEWIDTH+jet*JETWIDTH];
        im_test = in_test[rot+scale*SCALEWIDTH+jet*JETWIDTH+(JETWIDTH/2)];
        re_data = in_data[rot+scale*SCALEWIDTH+jet*JETWIDTH];
        im_data = in_test[rot+scale*SCALEWIDTH+jet*JETWIDTH+(JETWIDTH/2)];

        mag_test = sqrt(pow(re_test,2) + pow(im_test,2));
        mag_data = sqrt(pow(re_data,2) + pow(im_data,2));

        sum1 += mag_test * mag_data;
        sum2a += pow(mag_test,2);
        sum2b += pow(mag_data,2);
      }
    }
    similarity += sum1/sqrt(sum2a*sum2b);
  }

  similarity = similarity/JETS;

  char *enddir = strrchr(in_file,'\\');
  in_file = &enddir[1];
  char *endname = strrchr(in_file,'.');
  if(endname) endname[0] = '\0'; //terminate string
  strcat(in_file,".pgm");

  char *filename;
  filename = (char*) malloc(sizeof(char)*MAX_FILENAME_LENGTH);
  strcpy(filename, DEFAULT_PGM_DIR);
  strcat(filename,in_file);
```

```
  if(*out_similarity1 == NULL) {
    *out_similarity1 = similarity;
    *out_file1 = (char*) malloc(sizeof(char)*MAX_FILENAME_LENGTH);
    strcpy(*out_file1,filename);
    return MRV_SUCCESS;
  }

  if(similarity > *out_similarity1) {
    *out_similarity1 = similarity;
    *out_file1 = (char*) malloc(sizeof(char)*MAX_FILENAME_LENGTH);
    strcpy(*out_file1,filename);
    return MRV_SUCCESS;
  }
  return MRV_FAILURE;
}

REGISTER_MTIP_MODULE("Gabor_Recognize|Atle",GaborRecognize_MDF,GaborRecognize_MIF,0,MY
_HELP_FILE,MY_HELP_TOPIC);
```